Нижегородский государственный университет им. Н.И. Лобачевского

Национальный исследовательский университет

Учебно-научный и инновационный комплекс
"Новые многофункциональные материалы и нанотехнологии"

Основная образовательная программа
210100 «Электроника и наноэлектроника»,
222900 «Нанотехнологии и микросистемная техника»

Кудрин А.В.

ИСПОЛЬЗОВАНИЕ ПРОГРАММНОЙ СРЕДЫ LABVIEW ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕДЕНИЯ ФИЗИЧЕСКИХ ЭКСПЕРИМЕНТОВ

Электронное учебно-методическое пособие

Нижний Новгород
2014

ИСПОЛЬЗОВАНИЕ ПРОГРАММНОЙ СРЕДЫ LABVIEW ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕДЕНИЯ ФИЗИЧЕСКИХ ЭКСПЕРИМЕНТОВ. Кудрин А.В. Электронное учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2014. – 68 с.

Рецензент к.ф.-м.н., доцент В. В. Карзанов

В учебно-методическом пособии рассмотрены основы графического языка программирования G, используемого в программной среде LabView. Изложены основные особенности работы с данными различного типа, циклами, структурами последовательности, ветвления, преобразования типа данных. Описано взаимодействие программной среды LabView с реальными измерительными приборами. Рассмотрен пример программы для управления экспериментальной установкой.

Электронное учебно-методическое пособие предназначено для студентов ННГУ, обучающихся по направлению подготовки 210100 «Электроника и наноэлектроника» и 222900 «Нанотехнологии и микросистемная техника».

Оглавление

Введение.

Методическое пособие посвящено рассмотрению основных вопросов по организации автоматизации проведения физических экспериментов с использованием программной среды LabView. Научно-исследовательская деятельность физика-экспериментатора основывается на проведении разнообразных экспериментов с целью получения данных об определенных свойствах объекта исследования и последующей обработке полученных данных. В настоящее время большое распространение получают аппаратно-программные комплексы, позволяющие проводить изучение определенного эффекта с максимальной степенью автоматизации и с минимальным влиянием экспериментатора на ход выполнения эксперимента. В предельном случае задача исследователя сводиться к помещению исследуемой структуры в экспериментальную установку и выбора параметров эксперимента из предложенных вариантов. Примерами таких экспериментальных установок являются: установки по исследованию эффекта Холла, установки вольт-фарадного электрохимического профилирования, установки картирования фотолюминесценции, сканирующие зондовые микроскопы, рентгеновские дифрактометры, электронные микроскопы, магнитометры. В то же время большая часть экспериментальной работы проводиться на исследовательских комплексах, созданных непосредственно коллективом исследователей. Это позволяет использовать наиболее подходящие или доступные компоненты для максимально эффективного решения поставленной задачи. Оптимизация исследовательского комплекса позволяет добиться решения конкретной экспериментальной задачи. Для создания подобного исследовательского комплекса используются отдельные приборы и компоненты, такие как вольтметры, мультиметры, синхронные детекторы, детекторы излучения, аналого-цифровые и цифро-аналоговые преобразователи, источники питания и др. Современные подобные приборы имеют возможность удаленного управления и считывания данных с помощью компьютера, это позволяет

создать аппаратно-программный экспериментальный комплекс для эффективного проведения эксперимента. Одним из возможных способов создания программной части экспериментального комплекса является использование программной среды LabView. Использование LabView имеет ряд преимуществ над другими программными средами, а именно:

1) Используемый в LabView графический язык программирования G прост для освоения людьми, имеющими базовые знания по программированию и не имеющими значительного опыта программирования. Наглядное графическое представление потока данных в программе и основных структурных элементов программы позволяет относительно легко создавать и отлаживать программы по управлению прибором.
2) В виду изначальной ориентации LabView на использование в системах сбора и обработки данных, в LabView иметься богатый набор отлаженных инструментов для установки связи с реальными приборами посредством различных коммуникационных интерфейсов, таких как GPIB, COM, USB, LAN, PCI. Большинство современных приборов оснащенных коммуникационными интерфейсами поставляются с драйверами, являющимися подпрограммами LabView, которые могут быть использованы в создаваемой программе и обеспечивают легкое управление прибором на высоком уровне. Кроме того в LabView возможно установление управления прибором на низком уровне, например, посредством передачи команд, соответствующих стандарту SCPI.
3) ННГУ является обладателем лицензии на программную среду LabView, что делает доступным ее использование для автоматизации исследовательских и образовательных установок.

Целями учебно-методического пособия являются:

1) Ознакомить научных сотрудников, преподавателей, аспирантов и студентов с доступным способом создания автоматизированных

исследовательских установок. Пособие в частности будет полезно сотрудникам и студентам физического факультета ННГУ и НИФТИ ННГУ.

2) Наиболее полное использование в научной работе и учебном процессе оборудования, закупленного по программе НИУ, поскольку на его основе могут быть созданы автоматизированные экспериментальные комплексы, оптимально соответствующие поставленной задаче. В частности учебные измерительные платформы National Instruments (NI) ELVIS 2 и приборные комплексы на основе платформы NI PXI (полностью совестимые с NI LabView) поставляются с узкоспециализированным программным обеспечением, однако с соответствующим программным обеспечением могут использоваться в широком круге задач.

Задачи учебно-методического пособия:

1) Ознакомление с основами графического языка G: типы данных, потоки данных, циклы, массивы. Рассмотрение основных программ по сбору данных с прибора и передачи команд прибору, использование команд стандарта SCPI и драйверов приборов для LabView.
2) Ознакомление с особенностями управления приборами посредством программной среды LabView. Ознакомление с программным интерфейсом VISA между аппаратной платформой и программной средой. Рассмотрение основных особенностей установления связи между компьютером и прибором посредством интерфейса VISA с использованием наиболее распространенных аппаратных интерфейсов COM и USB.
3) Рассмотрение примера программы для автоматизации экспериментальной установки измерения намагниченности, используемой в НИФТИ ННГУ

Предлагаемое пособие состоит из трех двух глав. В первой главе изложены основы графического языка G. Во второй главе рассмотрены особенности удаленного управления прибором посредством компьютера с использованием программной среды LabView, способы передачи команд

прибору и считывания данных. В третьей главе рассмотрены особенности управления экспериментальной установкой состоящей из нескольких приборов.

1. Знакомство с основами программирования в LabView.

1.1. Основные операции с данными числового и логического типа. Циклы.

В общем случае программа управляющая работой экспериментальной установкой должна выполнять следующие основные функции:

1) Возможность задачи исходных параметров эксперимента, такие как режим функционирования приборов, величины генерируемые приборами, диапазоны измерений, первоначальные значения и диапазоны изменения параметров, варьируемых в процессе эксперимента.
2) Изменение варьируемых параметров эксперимента согласно заданному алгоритму. Сбор и накопление данных, регистрируемых в ходе выполнения эксперимента. Визуализация хода проведения эксперимента, т.е. построение исследуемой зависимости в ходе выполнения эксперимента.
3) Сохранение массива полученных данных для последующей обработки.

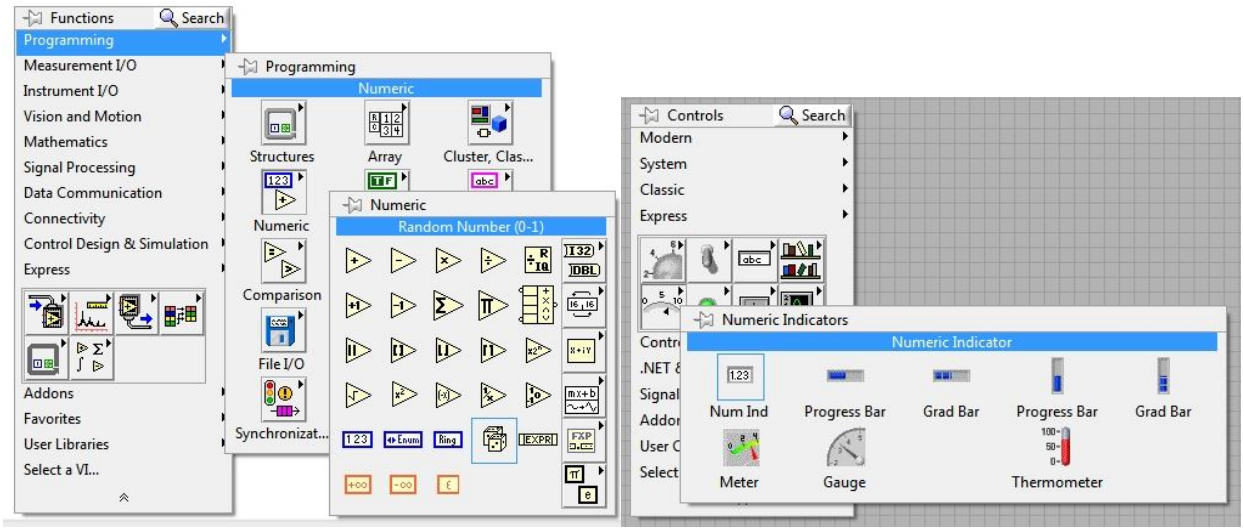
В простейшем случае программа по автоматизации эксперимента может выполнять функции, указанные в пунктах 2 и 3. Например, режим работы прибора можно не задавать удаленно, а выбрать с помощью органов управления прибором (если они присутствуют). В этом случае основной задачей программы является сбор и сохранение регистрируемых данных. Предположим, что измерительный прибор регистрирует некоторую изменяющуюся величину и основной задачей является считывание текущего значения величины и сохранение массива данных, содержащих ее значения.

Вначале смоделируем в LabView работу прибора в таком режиме. После запуска LabView выберем пункт соответствующий новому проекту (Blank VI). Каждый проект (или виртуальный инструмент VI- virtual instrument) состоит из окна интерфейса приложения (передняя панель - Front

Panel) и окна программного кода (Block Diagram). Переключение между окнами расположено в пункте Windows основного меню окон (Show Block Diagram/Show Front Panel).

В качестве модели источника измеряемой величины воспользуемся генератором случайных чисел. Перейдем в окно программного кода. Нажатием правой кнопки мыши вызовем меню основных функций (Functions, рис. 1 а). Далее в пункте Программирование (Programming) зайдем в пункт операций с числами (Numeric) и нажмем на иконку генератора случайных чисел (игральные кости). После этого в любой области окна можно поместить графический объект соответствующий программному генератору случайных чисел[I]. Для вывода сгенерированных чисел перейдем на переднюю панель и нажатием правой кнопки мыши вызовем основное меню объектов (Controls), какие можно разместить на передней панели. В пункте Express найдем вкладку соответствующую индикатору числовых типов данных (Numeric Indicators) и выберем индикатор Num Ind, отображающий данные в виде десятичной дроби (рис. 2 б). Индикатор числовых типов данных может иметь и другой вид, например, в виде заполняющейся полоски (Progress Bar) или аналогового индикатора (Meter) и др. Вернувшись в окно программного кода, увидим, что индикатору Num Ind в программном коде соответствует символ (элемент) с названием Numeric. Функции, константы, переменные и другие элементы в языке G представляются в виде графического объекта с участками (терминалами) для подключения потока данных. Терминал может быть входным (для ввода данных в элемент) и выходным (для вывода данных). У индикатора имеется один терминал (входной) расположенный слева на символе.

[I] По нажатию правой кнопки мыши на элемент в области программного кода, соответствующей некоторой функции, появляется контекстное меню, содержащее пункт Help. При выборе этого пункта появляется информативная справка программной среды LabView, содержащее подробное описание функции и ссылки на встроенные примеры использования функции.



а) б)

Рисунок 1. а - Меню основных функций, подменю программирования и операций с числами в окне программного кода. б - Меню основных элементов передней панели и подменю индикаторов численной переменной.

При подведении курсора к левой стороне иконки генератора случайных чисел появляется оранжевый кружек, соответствующий участку выхода данных (выходному терминалу), а курсор приобретает вид катушки с проводом. Удерживая нажатой левую кнопку мыши необходимо соединить проводником выходной терминал генератора с входным терминалом индикатора (рис. 2 а). Проводник является графическим символом потока данных из генератора случайных чисел в индикатор на лицевой панели.

Для запуска программы необходимо нажать на белую стрелку на лицевой панели либо на панели программного кода. После однократного нажатия произойдет генерация случайного числа в диапазоне от 0 до 1, а индикатор на передней панели отобразит сгенерированное число (рис. 2).

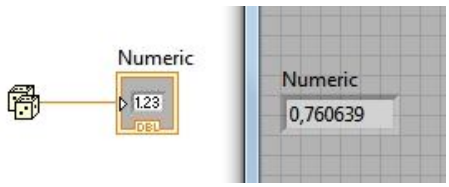


Рисунок 2. Часть окна программного кода (слева) и лицевой панели (справа) простейшей программы для генерации случайных чисел.

Как будет рассмотрено ниже, реальный измерительный прибор будет передавать измеренные значения однократно, после каждого обращения к прибору. Следовательно, в ходе реального эксперимента необходимо много раз обращаться к прибору для получения данных. Количество обращений в общем случае определяется:

А) заранее заданным числом измерений (шагов);

Б) интервалом времени между обращениями и, соответственно, временем проведения измерений.

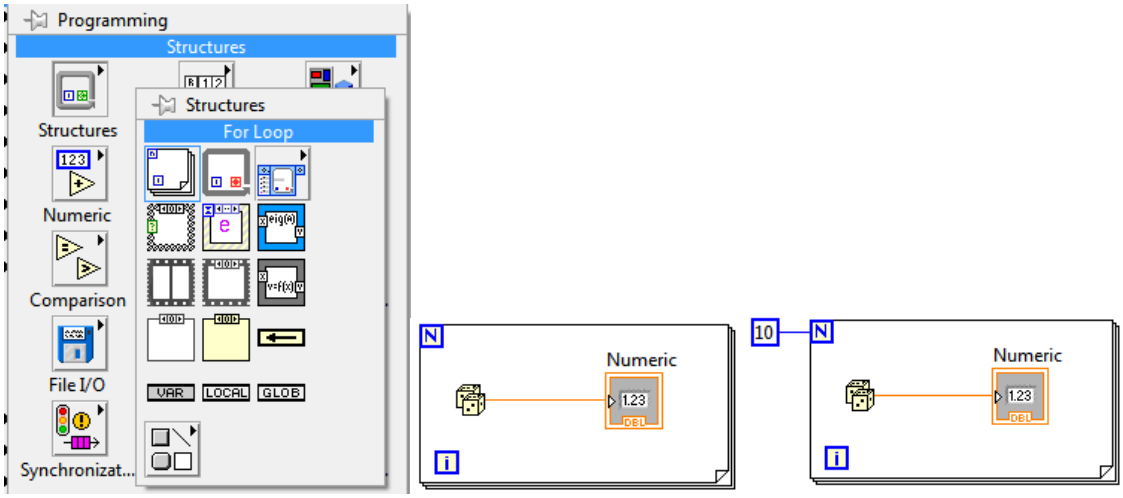
Поскольку необходимо последовательное проведение определенного числа однотипных операций (отправка запроса на прибор, получение ответа, преобразование типа данных, добавление данных в массив), то для их автоматического выполнения необходимо использовать структуры циклов. В программной среде LabView существует два вида циклов:

- цикл с фиксированным числом итераций For Loop;
- цикл по условию While Loop.

Графические элементы циклов расположены в подпункте Структуры (Structures) пункта Программирование основного меню, вызываемого в окне программного кода, как было рассмотрено выше (рис. 1 а). Для случая А), когда число выполняемых программой итераций определяется заранее, удобно использовать цикл For. Для случая Б) удобнее использовать цикл While.

Сделаем с помощью цикла For, чтобы генератор случайных чисел запускался заданное число раз. Выберем символ цикла For на вкладке

Структуры (рис 3 а). После нажатия левой кнопки мыши на поле программного кода появиться изображение четырехугольника, размерами которого можно управлять. Нарисуем этот четырехугольник поверх символа генератора и индикатора вывода числовых данных. Появиться белая область цикла For (рис 3 б) в котором будут размещены символы генератора, индикатора, терминала числа итераций (синий квадрат с буквой N) и счетчика итераций (синий квадрат с буквой i).

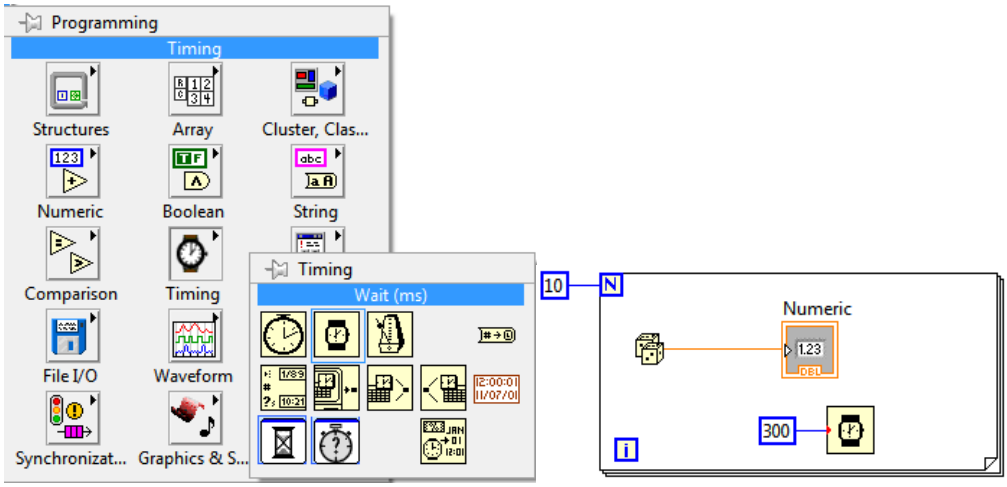


а) б) в)

Рисунок 3. а – меню элементов организации циклов и последовательностей. б – цикл с фиксированным числом шагов For.

Терминал числа операций (N) является входным. Для запуска цикла в этот терминал нужно подать целочисленное значение, соответствующие необходимому числу итераций цикла. Чтобы задать число итераций цикла нужно поместить целочисленную константу в окно программного кода. Целочисленная константа представляет собой синий прямоугольник и находиться во вкладке Numeric (рис. 1 а). В появившийся символ можно ввести любое целочисленное значение. Далее необходимо соединить выходной терминал константы с терминалом числа операций (N) (рис. 3 в). Синий соединительный провод и синие терминалы (символы) означает, что производятся операции с целочисленными данными. Желтый

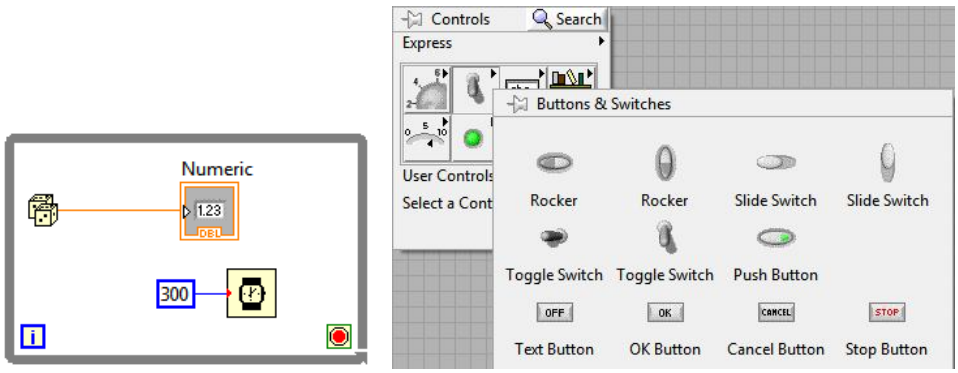
соединительный провод и желтые терминалы соответствуют данным с плавающей запятой. После запуска программы будет проведена генерация и отображения 10 случайных чисел. Однако на лицевой панели отобразиться только последнее из чисел, т.к. работа цикла будет проведена максимально быстро.



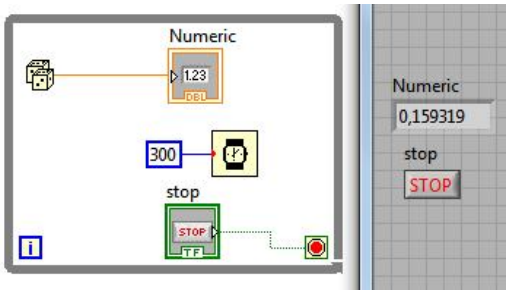
б) в)

Рисунок 4. а - меню элементов временных задержек и тактирования. б – Цикл For с заданной задержкой между итерациями.

Можно задать в явном виде временной интервал между итерациями цикла. Для этого в подпункте Timing пункта Программирование нужно выбрать элемент Wait (Ожидать) и поместить его внутрь цикла (рис. 4 а и б). Данный элемент делает задержку в выполнении участка кода программы в заданное количество миллисекунд. Значение задержки задается с помощью целочисленной константы, соединяемой с входным терминалом элемента Wait (рис. 4 б). Величина задержки задается в миллисекундах. Например, задержка в 300 мс позволяет после запуска программы увидеть последовательно 10 сгенерированных чисел.



а) б)



в)

Рисунок 5. а - цикл по условию While, б – элементы управления логическим типом данных, в - цикл по условию While и его элементы управления и индикации.

Заменим цикл с фиксированным числом итераций на цикл по условию While. Выберем символ цикла While на вкладке Структуры пункта Программирование (рис 3 а) и поместим внутрь структуры цикла генератор, индикатор и элементы задержки (рис. 5 а). В цикле While присутствует счетчика итераций (i) и элемент условия выхода из цикла (зеленый квадрат с красным кружком). Элемент условия выхода из цикла является входным терминалом для логического типа данных (True или False). По умолчанию цикл While работает непрерывно, пока на терминал условия выхода не поступит значение True. Однако сразу после создания цикла While программа запущена быть не может (стрелка запуска отображается в виде разорванной серой стрелки)[II]. Необходимо терминал условия выхода соединить с

[II] На передней панели рядом с кнопкой запуска в виде стрелки (Run) присутствует кнопка прерывания выполнения программы (Abort Execution). Такой способ остановки выполнения программы может использоваться при ее зависании.

источником логического типа данных. В качестве такого источника разместим на передней панели кнопку остановки цикла. Для этого в пункте Express на передней панели найдем вкладку Buttons&Switchers (кнопки и переключатели) и выберем элемент Stop Button (Кнопка остановки). Элементу управления Кнопка остановки на передней панели соответствует зеленый квадрат stop в области программного кода (рис. 5 в). Выходной терминал элемента stop является источником логического типа данных. Элемент stop необходимо поместить в область цикла While и соединить с терминалом условия выхода из цикла. Элемент stop и проводник данных являются зелеными, такой цвет соответствует логическому типу данных (рис. 5 в). Теперь после запуска программы будет каждые 300 мс происходить генерация случайного числа и отображение его в индикаторе на передней панели. После нажатия кнопки stop, этим элементом управления будет сгенерирована логическая переменная True. По зеленому проводнику из выходного терминала stop логическая переменная True будет передана в терминал условия выхода цикла While, что является условием остановки цикла.

Как упоминалось выше, циклы While и For имеют счетчик итераций цикла (синий квадрат с буквой i). Счетчик имеет выходной терминал, соответствующий целочисленным типом данных. Таким образом, можно узнать, сколько итераций было выполнено в ходе работы цикла до его остановки. Это может быть особенно полезно при работе с циклом While. Добавим на лицевую панель второй индикатор числовых типов данных. Соединим выходной терминал счетчика итераций цикла с входным терминалом индикатора. После запуска программы первый числовой индикатор будет отображать генерируемое случайное число, а второй индикатор отображает номер текущей итерации цикла.

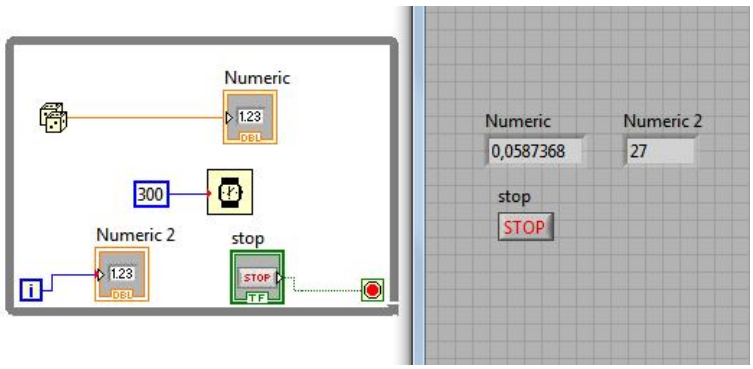


Рисунок 6. Часть окна программного кода (слева) и лицевой панели (справа) программы для генерации случайных чисел с помощью цикла While и отображения номера текущей итерации.

С помощью счетчика итераций цикла можно организовать выход из цикла по условию While. Например, пусть остановка цикла While производится при нажатии кнопки Stop Button либо при выполнении циклом определенного числа шагов, задаваемого на передней панели программы. Поместим на переднюю панель элемент управления данными числового типа Num Ctrl, представляющего собой окно в которое можно вводить число (в том числе в виде десятичной дроби) (рис. 7).

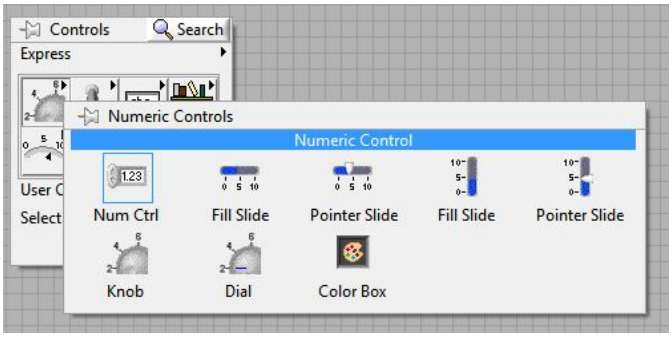
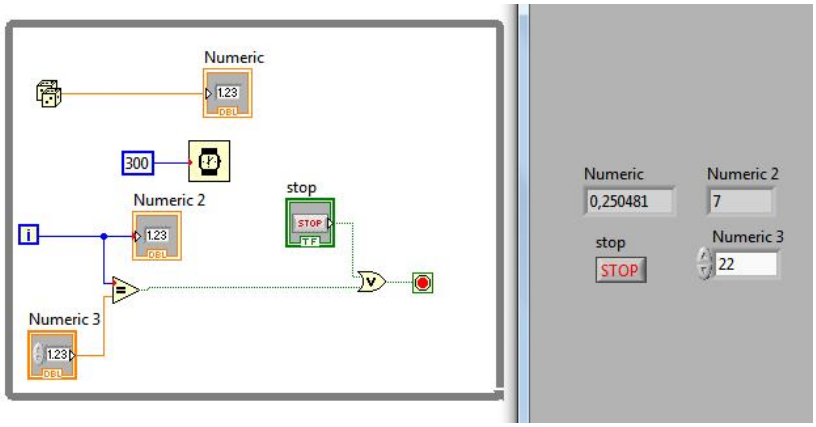


Рисунок 7. Меню основных элементов передней панели и подменю элементами управления данными числового типа.

Этот элемент управления позволит задать максимальное число итераций цикла While. Следует отметить, что индикатор данных числового типа (Num Ind) на передней панели имеет серый фон, в то время как элемент управления данными числового типа (Num Ctrl) имеет белый фон. Поскольку для

остановки цикла в терминал условия выхода цикла While должна быть подана логическая переменная True, необходимо добавить элемент генерации логической переменной при достижении циклом числа итераций, заданных в элементе управления Num Ctrl. Для этого можно использовать функцию сравнения Equal? Данная функция имеет графический элемент в виде треугольника со знаком равенства и может быть добавлена в области программного кода в подпункте операций сравнения (Comparison) пункта Программирование (Programming). Данный элемент имеет два входных терминала для данных числового типа и один выходной терминал для данных логического типа. В случае если два числа, которые поступают на входные терминалы элемента Equal? равны, на выходном терминале элемента генерируется логическая переменная True. Соединим выходной терминал счетчика итераций цикла с одним их входных терминалов элемента Equal? (рис. 8). Со вторым входным терминалом элемента Equal? соединим выходной терминал элемента управления данными числового типа Num Ctrl (рис. 8).



в)

Рисунок 8. Часть окна программного кода (слева) и лицевой панели (справа) программы для генерации случайных чисел с помощью цикла While, отображения номера текущей итерации и выхода из цикла по истечению заданного числа итераций.

Следует отметить, что не имеет значения к какому из двух входных терминалов подсоединяем проводники данных числового типа. Элемент Equal? генерирует переменную True при поступлении числа на один терминал, числу равному второму на другом терминале, независимо от порядка их поступления и номера входного терминала.

Теперь можем задавать на лицевой панели в окне управления Num Ctrl максимальное число шагов выполнения цикла, например 22 (рис. 8). При достижении числа итераций равного 22, элемент Equal? сгенерирует переменную True, которая может быть использована для остановки цикла. Однако, нам необходимо сохранить возможность остановки цикла с помощью Кнопки остановки, которая также генерирует переменную True. Следовательно, нужно ввести функцию логического сравнения ИЛИ, которая будет выдавать переменную True для остановки цикла в зависимости от того, что произошло – была нажата Кнопка остановки (элемент кнопки выдал True) или достигнуто максимальное число шагов (элемент Equal? выдал True). Элемент логического сравнения ИЛИ (Or) находится в области программного кода в подпункте операций с данными логического типа (Boolean) пункта Программирование (Programming). Элемент имеет два входных терминала для данных логического типа и один выходной (рис. 8). Соединим входные терминалы элемента Or с выходными терминалами Кнопки остановки и элемента Equal?, а выходной терминал с входным терминалом элемента остановки цикла (рис. 8). После запуска программы цикл While остановится после нажатия Кнопки остановки, либо после истечения 22 итераций цикла (рис. 8). Созданное дополнительное условие выхода из цикла может рассматриваться как аналог команды Break в других языках программирования.

Обратим внимание, что в рассматриваемом примере элемент Equal? сравнивает различные данные числового типа – сравнивается целочисленная переменная и переменная с плавающей десятичной запятой. Поэтому один входной проводник имеет синий цвет, второй желтый. При работе с данными

числового типа в языке программирования G нет необходимости проводить преобразование типа данных. Поскольку элемент управления Num Ctrl соответствует численным данным с плавающей десятичной запятой, в этот элемент управления можно ввести десятичную дробь, например 22,5 (в LabView десятичным разделителем является запятая). Программа запустится, но остановка цикла будет возможна только после нажатися Кнопки остановки, т.к. текущее число шагов (целочисленная переменная) очевидно никогда не станет равным заданной десятичной дроби. В LabView возможно задать форму представления данных числового типа. Например, после нажатия правой кнопки мыши на элементе управления Num Ctrl появляется контекстное меню, содержащее пункт Properties, содержащая вкладку Data Type. По умолчанию числовые данные элемента Num Ctrl представляются в форме Floating-point double-precision (т.е. числа с плавающей десятичной запятой двойной точности). В подпункте Representation пункта Properties можно выбрать иное представление числовых данных, например Word signed integer (целые числа в диапазоне от – 32768 до +32768). Теперь в области программного кода элемент Num Ctrl отображается в виде синего квадрата, в соответствии с цветом целочисленных данных числового типа. В этом случае в элемент управления Num Ctrl можно ввести только целые числа. В случае ввода дробного числа, число округляется до ближайшего целого.

1.2. Отображение и сохранение данных. Массивы.

Одной из основных задач программы управляющей экспериментальной установкой является сохранение данных, регистрируемых в ходе проведения эксперимента. В нашей текущей программной модели источника измеряемой величины присутствуют два набора меняющихся данных – номер текущей итерации цикла и величина, выдаваемая генератором случайных чисел. Следовательно, после выполнения, например, десяти итераций цикла будет проведена генерация

десяти случайных чисел. Очевидно, что на каждом шаге цикла генерируется одно число, соответствующее данной итерации. Таким образом, в ходе выполнения программы может быть создан массив данных, содержащий все случайные числа, созданные генератором случайных чисел. Можно создать массив данных, содержащий числа, создаваемые счетчиком итераций цикла. Это будут одномерные массивы. Можно создать массив в котором каждому номеру итерации цикла будет соответствовать случайное число, сгенерированное в ходе этой итерации. В этом случае получаемый массив будет двумерным, т.е. матрицей. В этой матрице будет две строки: одна строка соответствует числам, созданным генератором случайных чисел, вторая строка соответствует числам, созданным счетчиком итераций. Число столбцов в матрице определяется числом итераций.

Рассмотрим простейший случай одномерного массива из случайных чисел. Создадим на передней панели элемент отображения, в котором будут представлены данные создаваемого массива. Для этого на переднюю панель нужно поместить шаблон массива Array (рис. 9).

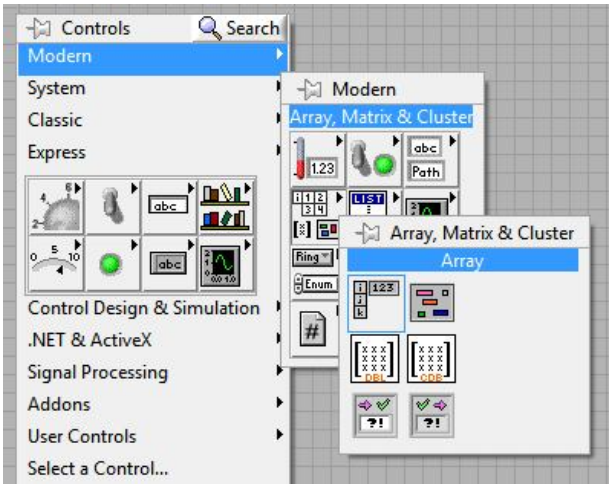
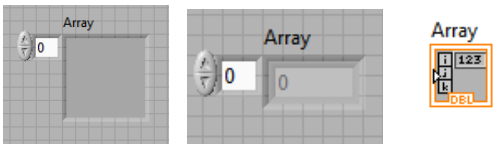


Рисунок 9. Меню элементов раздела Modern передней панели и подменю элементами управления массивами и матрицами.

Шаблон массива Array расположен во вкладке Array, Matrix @ Clusters в разделе Modern на передней панели и вызывается нажатием правой кнопки

мыши. После добавления на переднюю панель шаблон массива Array отображается в виде серого квадрата, в который необходимо поместить тот элемент, из которых будет состоять массив. Создадим массив из элементов отображения данных числового типа. Для этого создадим элемент отображения Num Ind (рис. 1 б) и поместим его внутрь серого квадрата элемента Array. После этого элемент Array на передней панели примет вид, представленный на рис. 10 б, а в области программного кода возникнет элемент Array с входным терминалом (рис. 10 в).



а) б) в)

Рисунок 10. а - шаблон массива Array, б – массив элементов отображения данных числового типа, в – графический символ программного элемента для вывода массива данных числового типа.

Таким образом, будет создан пустой массив данных числового типа и соответствующий ему элемент отображения на передней панели программы. Если поместить графический символ программного элемента для вывода массива данных числового типа вовнутрь цикла While и соединить с выходным терминалом генератора случайных чисел, то соединительный проводник отобразиться в виде пунктирной линии с разрывом (рис. 11).

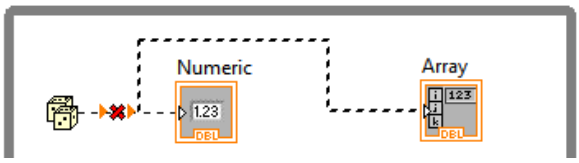
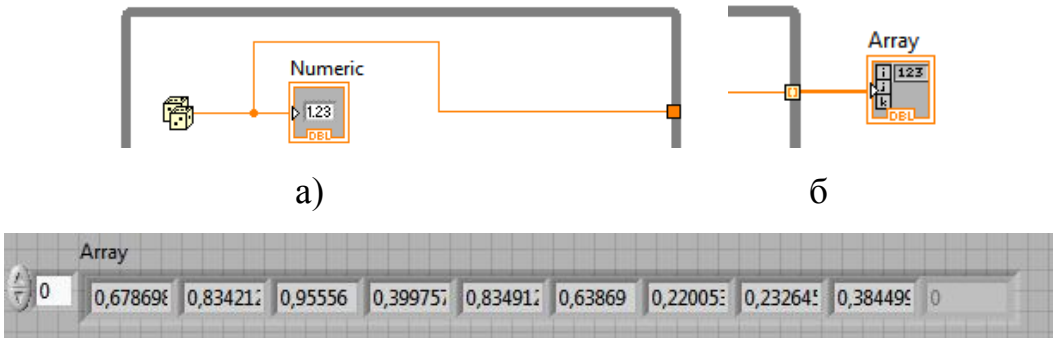


Рисунок 11. Часть окна программного кода для заполнения массива случайными числами.

Это связано с тем, что на вход элемента отображения массива необходимо подавать именно массив данных, в то время как генератор случайных чисел на каждом шаге цикла выдает одно число. Наиболее простым способом заполнения массива полученными данными является накопление данных на границе массива (атоиндексирование auto-indexing) и передача данных в массив после окончания выполнения цикла.

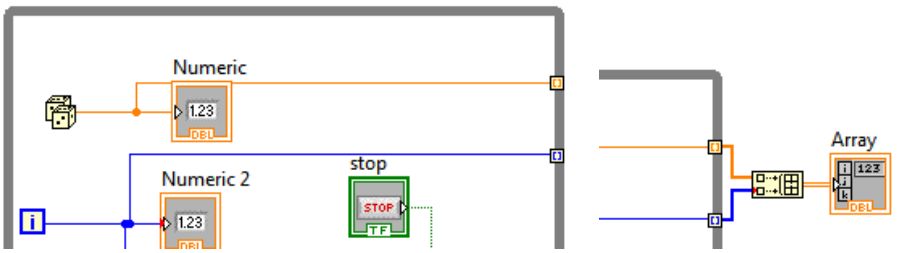


в)

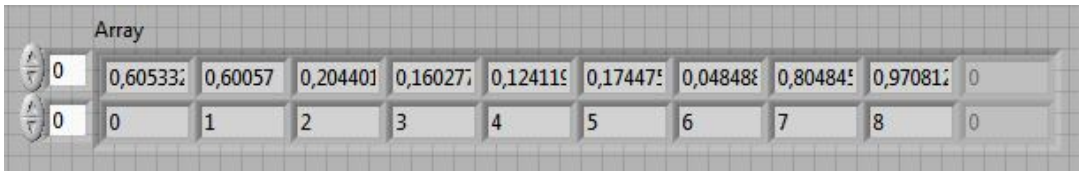
Рисунок 12. Часть окна программного кода и лицевой панели программы для заполнения массива случайными числами: а – автоматическая индексация данных на границе цикла выключена, б – автоматическая индексация данных на границе цикла включена, в – индикатор массива числовых данных.

Соединим выходной терминал генератора случайных чисел с правой границей цикла While (рис. 12 а). На границе появится символ в виде заполненного квадрата желтого цвета (цвет числового типа данных с плавающей запятой). Данный квадрат является выходным терминалом цикла. После окончания выполнения цикла на данном терминале появится число, равное числу с генератора случайных чисел на последней итерации цикла. Для накопления данных, полученных за все итерации цикла, необходимо выбрать пункт Enable Indexing в контекстном меню (по нажатию правой кнопки мыши на терминал) данного выходного терминала на границе цикла. После выбора автоиндексирования (Enable Indexing) терминал изменит свой

вид (рис. 12 б). Теперь после окончания цикла на данном терминале будет накоплен одномерный массив данных, состоящий из всех чисел, выданных генератором случайных чисел за все шаги цикла[III]. Соединим выходной терминал на границе цикла с ранее созданным элементом отображения массива данных числового типа (рис. 12 б). Соединительный провод при этом отображается в виде жирного желтого проводника. Увеличенная толщина проводник означает то, что по нему передается массив данных. Запустим программу. После окончания всех шагов цикла, в индикатор массива будут переданы все накопленные на границе цикла значения. На рис. 12 в представлен индикатор массива на передней панели после выполнения циклом While 9 итераций (отметим, что индикатор на передней панели растянут в горизонтальном направлении для отображения 10 ячеек). Можно видеть все 9 значений сгенерированных случайных чисел.



а) б



в)

Рисунок 13. Часть окна программного кода и лицевой панели программы для заполнения массива случайными числами: а – проводится автоматическая индексация данных в двух одномерных массивах на границе цикла, б – индикатор двумерного массива числовых данных.

[III] Отметим, что для массива For режим автоиндексирования включается по умолчанию при соединении проводника с данными с границей массива.

Теперь получим двумерный массив, содержащий номер шага цикла и сгенерированное на этом шаге случайное число. Соединим терминал счетчика итераций цикла с правой границей цикла и включим автоиндексирование для этих данных (рис. 13 а), аналогично тому, как было рассмотрено выше. Теперь на выходе из цикла имеется два одномерных массива. Теперь нужно их объединить в один двумерный массив и передать данные в индикатор двумерного массива. Для того чтобы создать индикатор двумерного массива нужно на передней панели индикатор Array растянуть вниз на одну ячейку. Нужно растянуть обе половины индикатора Array (левая половина является областью выбора элементов отображения, правая половина индикатора является непосредственно областью отображения). В этом случае индикатор становится индикатором двумерного массива числовых данных и в области программного кода к нему уже нельзя подсоединить проводник, соответствующий одномерному массиву.

Для объединения двух одномерных массивов в двумерный нужно в область программного кода добавить функцию Build Array (построить массив). Функция находится в области программного кода в меню основных функций по адресу Programming –> Array (меню функций и команд работы с массивами) –> Build Array. Данная функция объединяет несколько массивов в один массив более высокой размерности. Также с помощью этой функции можно в сформированный массив данных добавлять элементы без изменения размерности массива, как будет показано ниже. По умолчанию графический элемент, соответствующий этой функции, имеет один входной терминал и один выходной. Для увеличения количества входных терминалов необходимо растянуть графический элемент вниз. В рассматриваемом случае необходимо два входных терминала. Создав два входных терминала соединим их с выходными терминалами на границе цикла (с терминалами в которых проходит накопление элементов массива, рис. 13 б). Обратим внимание, что элемент Build Array может функционировать в двух режимах, случается так, что LabView автоматически выбирает режим для Build Array,

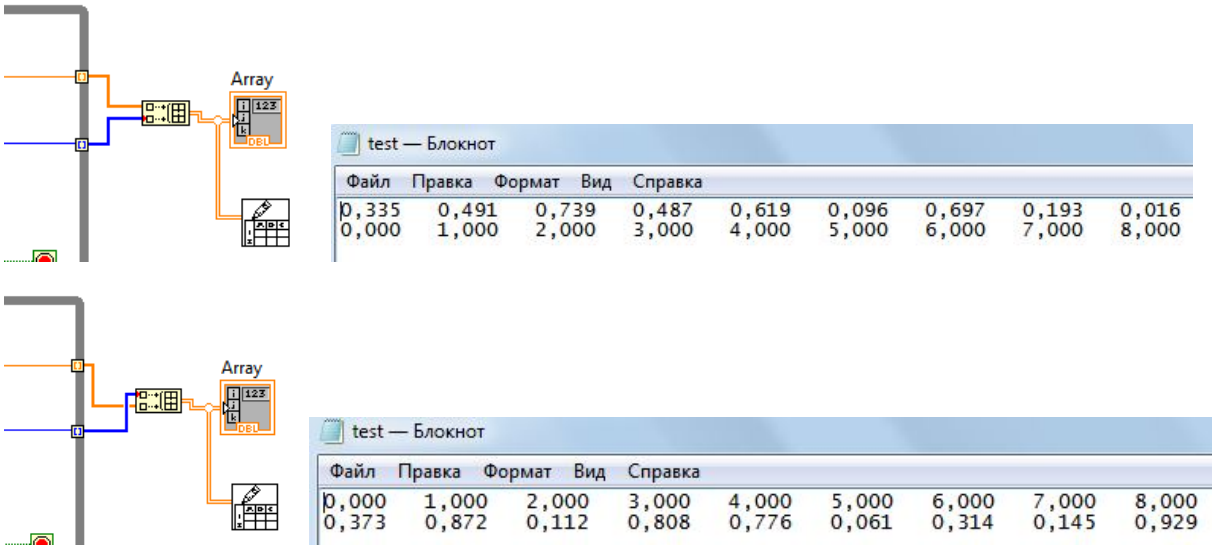
хотя в конкретном случае может быть необходим другой режим работы функции. По умолчанию функция Build Array работает в режиме повышения размерности выходного массива. В это случае, если на один входной терминал подается одномерный массив {1, 2} а на второй одномерный массив {3, 4, 5}, то на выходном терминале функции будет сформирован двумерный массив {{1, 2,}, {3, 4, 5}}. Этот режим необходим для рассматриваемого примера по заполнению массива случайными числами. Функция Build Array может находиться в режиме Concatenate Inputs (выбирается в меню по нажатию правой кнопки мыши на графический элемент). В этом случае функция просто объединяет входящие массивы в один без изменения размерности массива, т.е. на выходе создается одномерный массив {1, 2, 3, 4, 5}. Проследим, чтобы функция Build Array не находилась в режиме Concatenate Inputs и соединим выходной терминал функции с входным терминалом графического элемента индикатора Array (рис. 13 б). Обратим внимание, что проводник, соединяющий элементы Build Array и Array имеет вид широкой бело-желтой полосы, такой проводник символизирует передачу многомерного массива. После запуска рассматриваемой программы будет сформирован двумерный массив состоящий из случайных чисел и номера итерации массива While (рис. 13 в).

Следует отметить, что в большинстве случаев работы с массивами нет необходимости использовать элемент отображения массива (индикатора Array). В рассмотрено примере индикатор Array был введен для отображения созданного массива. Индикатор Array (как и элемент управления массивом Array) в языке программирования G не объявляет массив, т.е. для работы с массивом нет необходимости его объявляет.

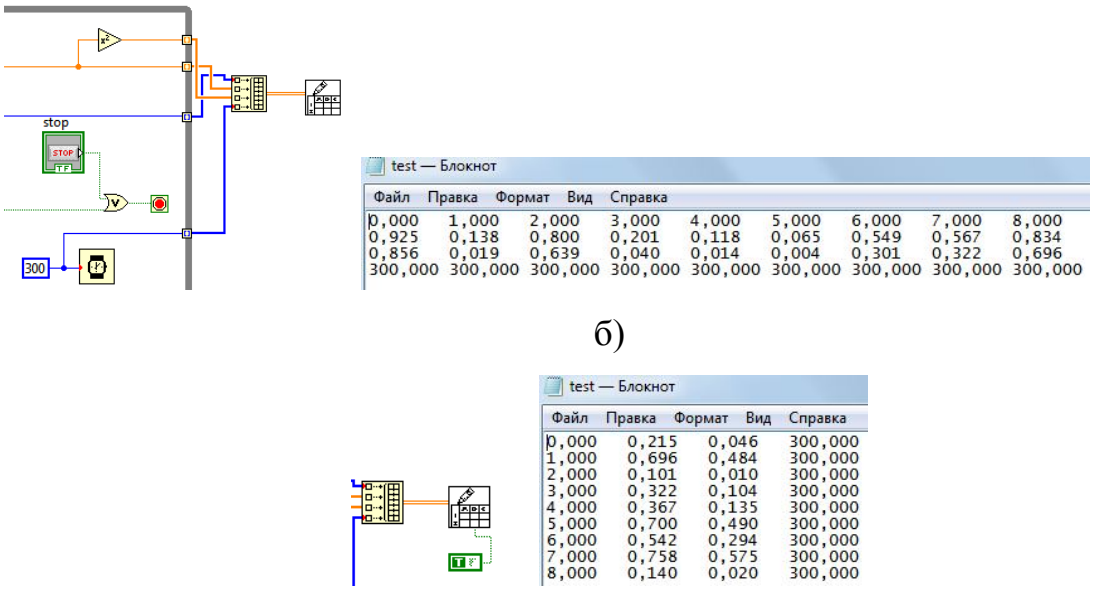
При работе программы управляющей реальной экспериментальной установкой присутствует необходимость сохранения полученных данных в файл для дальнейшей обработки полученных данных. Рассмотрим один из вариантов сохранения полученного двумерного массива данных в файл. Наиболее простым способом сохранения данных числового типа в файл

является использование встроенной функции сохранения Write To Spreadsheet File (Запись в табличный файл). Функция находится в области программного кода по адресу Programming –> File I/0 (меню функций ввода вывода файлов) –>Write To Spreadsheet File.

Функция имеет несколько входных терминалов разного типа и один выходной. Описание назначения всех терминалов можно посмотреть в справке LabView. Нажмем два раза левой кнопкой мыши на графический элемент функции Write To Spreadsheet File. При этом произойдет появление фронтальной панели нового проекта LabView с названием Write To Spreadsheet File (DBL).vi. Это является следствием того, что функция Write To Spreadsheet File является отдельным полноценным проектом LabView, созданным для сохранения данных в файл, т.е. является виртуальным прибором (как и рассматриваемый нами проект) и имеет соответствующее расширение .vi. Можно увидеть программный код этого виртуального прибора выбрав в пункте Windows основного меню окон подпункт Show Block Diagram. Подобный отдельный проект, имеющий собственное функциональное назначение, может использоваться в любом другом проекте LabView и называется виртуальный подприбор. Возможно создание собственных виртуальных подприборов для использования в сложных проектах. В этом случае упрощается структура программного кода проекта.



а)



б)

в)

Рисунок 14. Часть окна программного кода программы для заполнения массива случайными числами и сохранения в файл: а – сохранение в файл данных двумерного массива их двух строк, б – сохранение в файл данных двумерного массива их четырех строк, в – сохранение в файл данных двумерного массива в виде столбцов.

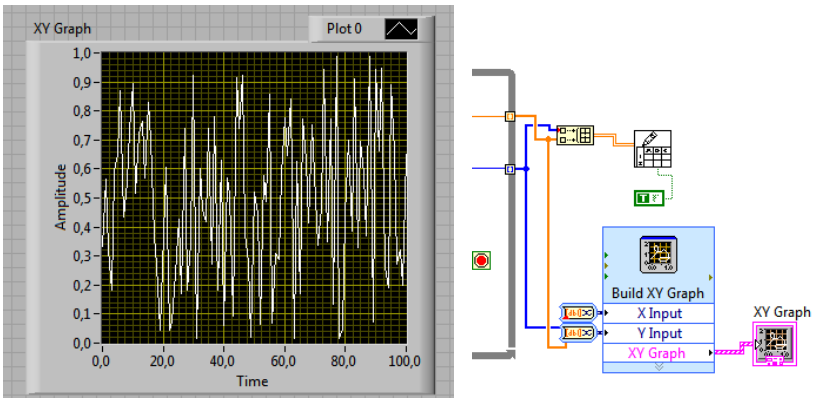
Соединим проводник с данными двумерного массива со входным терминалом 2D Data функции Write To Spreadsheet File. Этот терминал предназначен для подачи данных двумерного массива. Так же имеется терминал 1D Data для данных одномерного массива. Запустим

рассматриваемый проект. После его остановки (одним из двух ранее рассмотренных способов, например, после выполнения заданного числа шагов) появиться диалоговое окно Choose file to write в котором нужно будет выбрать путь для сохранения файла и указать название файла. Можно также указать расширение файла, например .txt. Файл представляет собой текстовый файл с двумя строками. Порядок строк определяется порядком вхождения одномерных массивов на входные терминалы функции Build Array (рис. 14 а). Двумерный массив данных может содержать более двух срок. Увеличение строк в массиве проводиться путем добавления входных терминалов функции Build Array. На рис. 14 б представлено формирование двумерного массива, состоящего их строки номера итерации цикла, строки случайных чисел, строки квадрата случайных чисел и строки с величиной задержки между тактами цикла (константа). При этом формируемый массив является двумерным, что позволяет подавать этот массив на вход 2D Data функции Write To Spreadsheet File (рис. 14 б).

Часто полученные данные двумерного массива удобнее представлять в виде столбцов. Поскольку данные двумерного массива, поступающие в функцию Write To Spreadsheet File являются матрицей, поэтому для отображения данных в виде столбцов нужно транспонировать сохраняемую матрицу. Для этого у функции сохранения в файл имеется входной терминал transpose? для логического типа данных. При подключении к этому терминалу логической константы True (Programming –> Boolean –> True), происходит транспонирование матрицы сохраняемого двумерного массива (рис. 14 в).

Получаемые в ходе выполнения данные можно представить графически в виде зависимостей одних данных от других. Для рассматриваемого примера отобразим графически зависимость величины случайного числа от номера этого числа (т.е. от номера итерации цикла). В LabhView имеется виртуальный подприбор для простого отображения величины Y (в виде данных одномерного массива) от величины X (также в

виде одномерного массива). Этот элемент отображения XY Graph находится в подпункте Graph Indicators пункта Express меню Controls на передней панели.



а) б

Рисунок 15. Часть лицевой панели кода и окна программного кода программы для визуализации процесса заполнения массива случайными числами: а – элемент отображения XY Graph зависимости Y(X), б – виртуальный подприбор функции XY Graph.

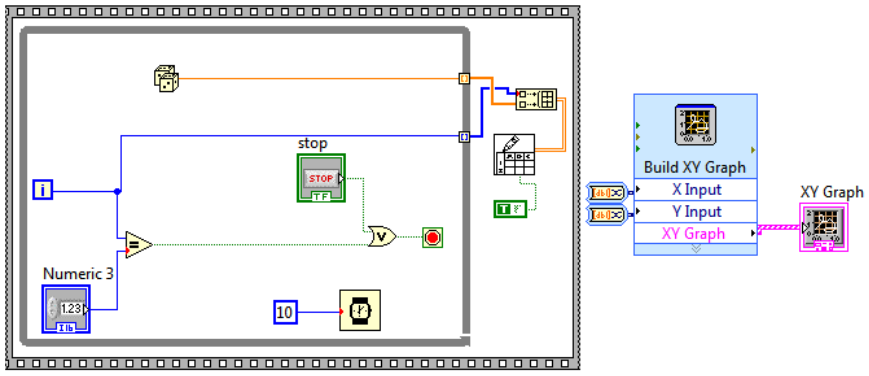
На рис. 15 представлен элемент отображения XY Graph на передней панели проекта. В области программного кода элементу XY Graph соответствуют два объекта: виртуальный подприбор Build XY Graph и непосредственно элемент отображения XY Graph (рис. 15 б). Подприбор Build XY Graph имеет входные и выходные терминалы. Подприбор Build XY Graph и индикатор XY Graph соединены толстым полосатым проводником (рис. 15 б). Проводники такого типа передают сложное объединение данных – кластеры. Кластер может содержать данные различного типа и различной размерности. В большинстве случаев достаточно использовать только два входных терминала подприбора Build XY Graph: терминал X Input (вход массива данных аргумента) и терминал Y Input (вход массива данных функции). Подведем к терминалу X Input массив номеров итераций цикла, а к терминалу Y Input массив случайных чисел (рис. 15 б). После завершения работы цикла появиться диалоговое окно Choose file to write, а индикатор XY

Graph отобразит зависимость величины случайного числа от номера случайного числа. На рис. 15 а представлена полученная зависимость после генерации 101 случайного числа. До запуска программы следует поставить нулевую (0 мс) задержку между итерациями цикла для максимальной скорости работы программы.

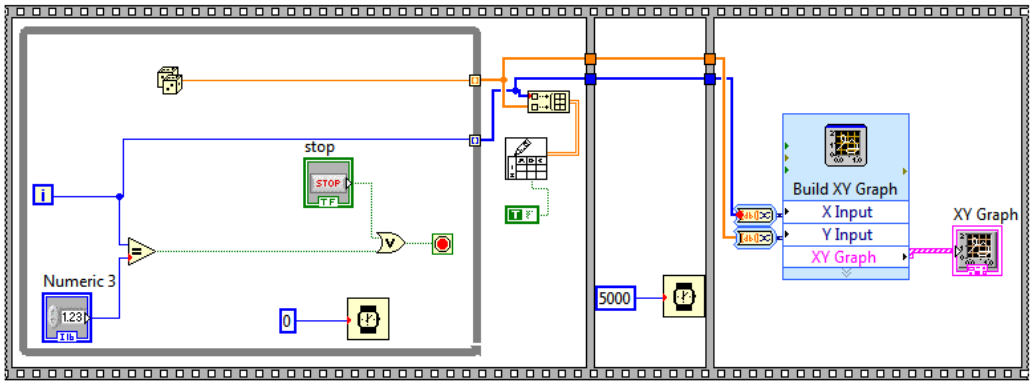
Обратим внимание на то, что после завершения работы цикла диалоговое окно Choose file to write и зависимость в индикаторе XY Graph появляются одновременно. Это ожидаемо, т.к. поток данных после структуры цикла приходит практически одновременно в элемент сохранения в файл Write To Spreadsheet File и индикатор отображения данных XY Graph. На самом деле поток данных в один из этих элементов приходит раньше, однако в общем случае неизвестно куда данные придут раньше. Данные в языке программирования G перемещаются слева направо. Данные по параллельным проводникам в области программного кода перемещаются независимо друг от друга и с различной скоростью, зависящей от сложности участка кода.

1.3. Структура последовательности. Сдвиговый регистр.

Нередко требуется точно определить последовательность выполняемых программой действий. Для этого можно использовать структуру Flat Sequence Structure (структура плоской последовательности). Такая структура находиться в подпункте Структуры (Structures) пункта Программирование основного меню в области программного кода. После выбора данной структуры необходимо очертить прямоугольник вокруг элементов программного кода, которые необходимо выполнить в первую очередь (рис. 16 а).



а)



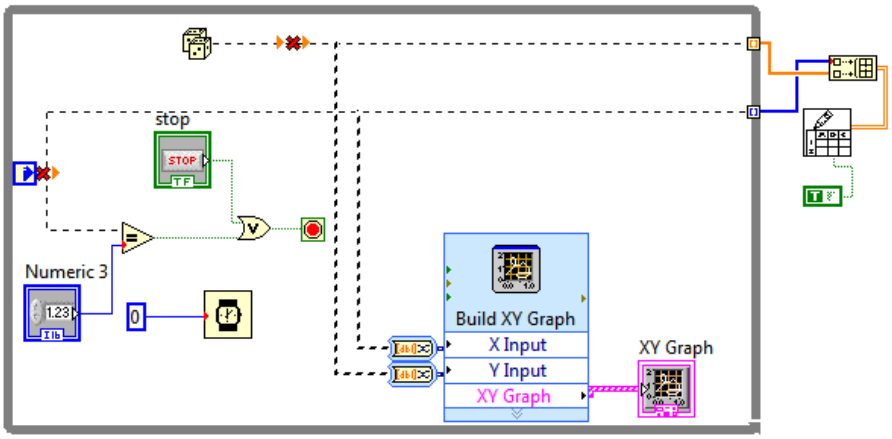
б)

Рисунок 16. Программа для визуализации процесса заполнения массива случайными числами: а – создание структуры последовательности действий, б – создание дополнительных кадров структуры последовательности.

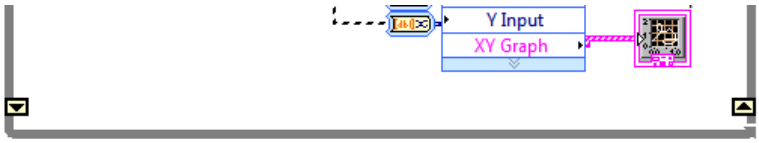
Структура последовательности состоит из выполняемых друг за другом кадров. После создания первого кадра можно добавить следующий кадр. Для этого в контекстном меню, появляющимся после нажатия правой кнопки мыши на правой границе первого кадра, нужно выбрать пункт Add Frame After. Добавим два кадра. Во второй кадр вставим элемент задержки Wait с временем ожидания 5000 мс (5 секунд), а в последний кадр вставим виртуальный подприбор Build XY Graph с индикатором XY Graph и соединим его с соответствующими источниками данных (рис. 16 б). При небольшом заданном числе итераций цикла часть программы, соответствующая генерации чисел и сохранения их в файл, будет выполнена очень быстро, однако отображение сформированного массива данных в

индикаторе XY Graph произойдет спустя 5 секунд, как было задано во втором кадре последовательности. Очевидно, что структуру последовательности можно использовать в различных случаях, когда нужно установить явный порядок выполнения участков программного кода.

В рассмотренном примере отображение результатов, получаемых в ходе выполнения программы, происходит фактически по окончанию основной части программы (рис. 15, 16). Т.е. график зависимости появляется после того, как сформирован весь массив полученных данных. Это связано с тем, что отображается массив данных, который был накоплен (индексирован) на границе массива, следовательно, эти накопленные данные можно использовать только после выхода потока данных из цикла по окончанию его работы. Наиболее удобным способом отображения получаемых/генерируемых данных является их визуализация в ходе выполнения работы. Это позволяет сразу оценить правильность выполнения программы или хода эксперимента. Очевидно, что для этого используемый виртуальный подприбор элемента отображения XY Graph нужно разместить внутри исполняемого цикла. Однако составная функция XY Graph отображает только поступающие на терминалы X Input и Y Input массивы данных и, поэтому нельзя подать на терминалы X/Y Input числовые данные (скаляры) формируемые в ходе каждой итерации цикла. Проводники с данными в этом случае будут разорваны, и программа не может быть выполнена (рис. 17 а). Следовательно, в ходе выполнения цикла необходимо формирование массивов данных, обращение к содержимому которых может быть выполнено на каждом шаге цикла. При этом содержимое массивов должно обновляться также при каждой итерации цикла.



а)



б)

Рисунок 17. Программа для визуализации процесса заполнения массива случайными числами: а – соединение функции XY Graph с источниками скалярных величин, б – добавление Сдвигового регистра в цикл.

Одним из способов создание таких массивов является использование особой функции циклов, такой как Сдвиговый регистр (Shift Register). Активация функции Shift Register выполняется путем выбора пункта Add Shift Register (добавить Сдвиговый регистр) в контекстном меню после нажатия правой кнопки мыши на боковых границах цикла. Сдвиговый регистр может быть создан как в цикле For, так и в While. Сдвиговый регистр необходим для передачи данных из одной итерации цикла в следующую итерацию. При этом в Сдвиговом регистре НЕ происходит сохранение (индексация) передаваемых данных. Сдвиговый регистр отображается в виде прямоугольников на границах цикла (рис. 17 б). Черный цвет прямоугольников означает, что первоначально тип передаваемых данных не определен. Каждый прямоугольник Сдвигового регистра имеет входной и выходной терминал. Сдвигового регистра работает следующим образом:

- в правом прямоугольнике сохраняются данные, которые были получены после завершения *текущей* итерации цикла;

- в левом прямоугольнике сохраняются данные, которые были получены после завершения *предыдущей* итерации цикла. Т.е. данные из правой части Сдвигового регистра передаются в левую часть и к ним можно обратиться.

Для упражнения посмотрим, как происходит работа Сдвигового регистра. Отсоединим от функции XY Graph проводники скалярных величин. Добавим на переднюю панель два индикатора данных числового типа. Подсоединим к входному терминалу правой части сдвигового регистра проводник, содержащий номер итерации цикла (рис. 18). Соединим также этот проводник и одним из индикаторов. При этом элементы сдвигового регистра стали синего цвета, что соответствует целочисленным цифровым данным. Второй индикатор соединим с выходным терминалом левой части сдвигового регистра. Теперь в ходе выполнения программы один из индикаторов будет отображать номер текущей итерации, а другой номер предыдущей итерации.

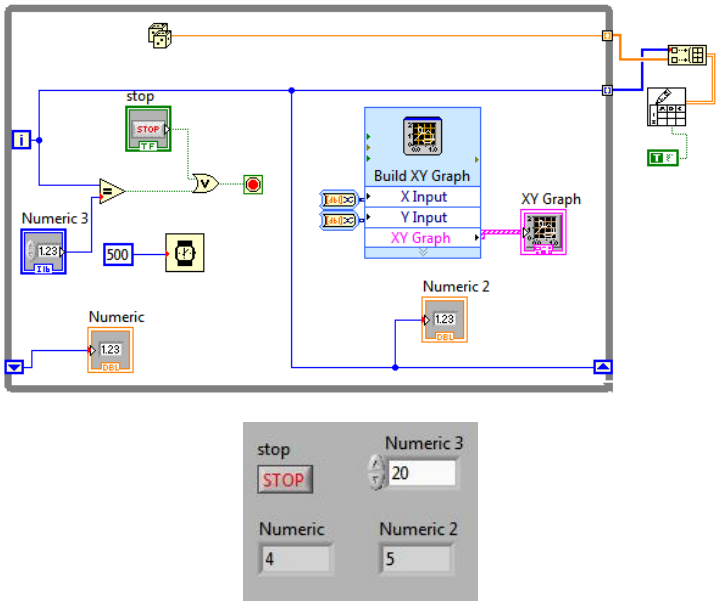
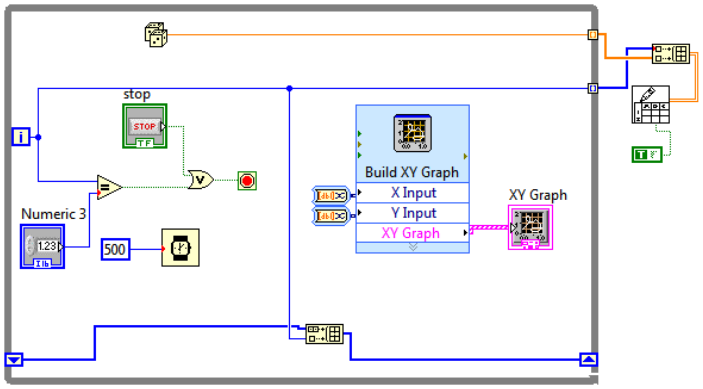
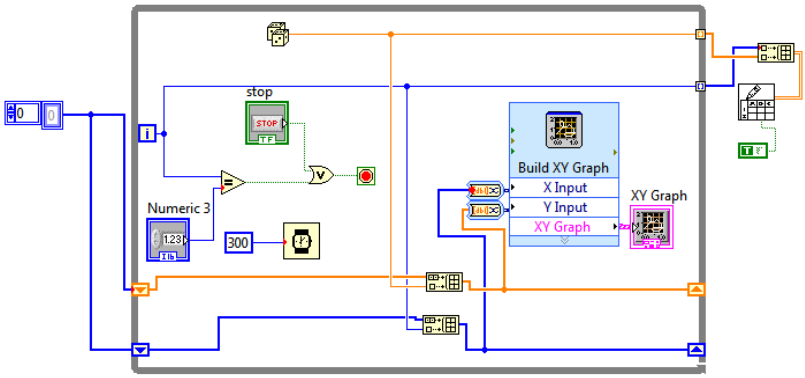


Рисунок 18. Работа сдвигового регистра. Слева окно программного кода, справа часть передней панели программы.

Для нашего случая (отображение данных в ходе работы цикла) нужно работать с массивом данных, который бы пополнялся в ходе работы цикла. Для этого нужно сначала объявить, что Сдвиговый регистр будет работать с массивом данных.



а)



б)

Рисунок 19. Работа сдвигового регистра: а – создание обновляемого одномерного массива данных, б – отображение получаемых данных в ходе выполнения цикла.

Добавим внутрь цикла элемент создания и добавления элементов в массив Build Array (рассмотрен выше). Растянем элемент Build Array вниз, чтобы у него было два входа. Выходной терминал (справа) элемента Build Array будет выдавать массив данных, соединим его с правой частью Сдвигового регистра (рис. 19 а). Во входной верхний терминал элемента Build Array нужно подать поток данных от массива, в который будут добавляться новые

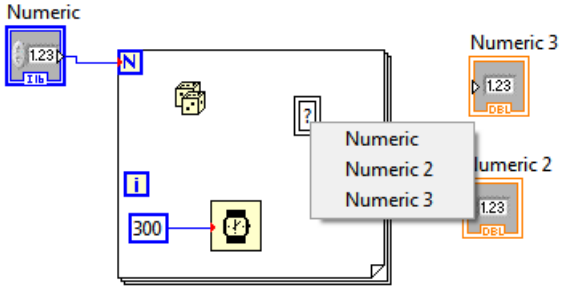
элементы, поэтому соединим этот терминал с левой частью сдвигового регистра. К входному нижнему терминалу элемента Build Array подведем источник скалярной величины (номер итерации), которая будет добавляться как очередной элемент массива в ходе выполнения цикла. После соединений, части Сдвигового регистра будут соединены с элементом Build Array толстыми проводниками, т.к. по ним передается одномерный массив, состоящий из номеров итераций цикла. Добавим в цикл еще один Сдвиговый регистр и аналогично организуем формирование массива данных из генерируемых случайных чисел (рис. 19 б). Теперь в ходе выполнения цикла в массивы добавляются новые элементы, затем массивы целиком передаются в новый шаг цикла, где в них добавляются новые элементы. Следовательно, внутри цикла стали доступны два массива с обновляющимся содержимым. Соединим выходной терминал элемента Build Array (фактически правую часть сдвигового регистра) с соответствующим терминалом (X Input и Y Input) элемента XY Graph (рис. 19 б). Поставим задержку Wait в несколько сот миллисекунд и запустим программу. Теперь видно, что зависимость величины случайного числа от номера итерации цикла отображается постепенно, в ходе выполнения цикла. После повторного запуска программы вновь создаваемые данные будут добавляться к значениям в массивах, уже сохраненных в памяти компьютера. Во избежание этого, нужно подавать пустой массив на вход левой части Сдвигового регистра. Для этого в области программного кода нужно добавить шаблон массива Array Constant (Programming –> Array–> Array Constant) и вставить в него нулевую целочисленную константу (рис. 19 б). Теперь при каждом новом запуске программы в Сдвиговые регистры передаются пустые одномерные массивы данных числового типа, далее они заполняются в ходе работы цикла.

1.4. Локальные переменные.

В ранее рассмотренных примерах данные передавались по проводникам, соединяющим структурные элементы программного кода. В связи с этим, данные формировавшиеся внутри цикла можно было использовать только для отправки в какие-либо функции, расположенные внутри структуры цикла. Вывод данных за пределы структуры цикла был возможен только по окончанию его работы. Однако может возникнуть необходимость использования данных, формируемых внутри структуры некоторого цикла, в других участках программного кода, обрабатывающих эти данные, например в другом параллельно исполняемом цикле. Для этого можно использовать специальные функциональные элементы – Локальные переменные (Local Variable). В отличие от других языков программирования, в языке G не нужно предварительно объявлять переменные и указывать их тип. Локальные переменные (как и глобальные) могут и не вводиться.

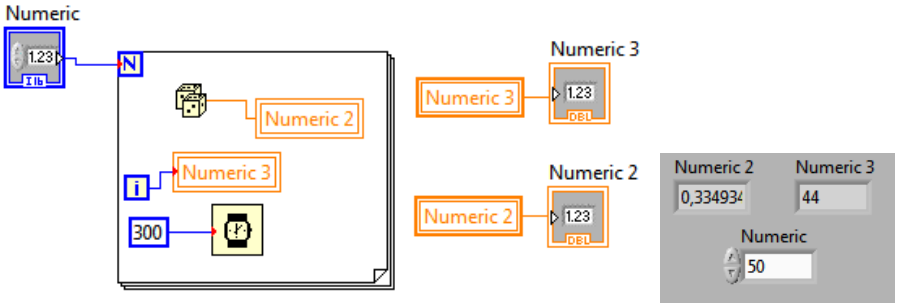
Рассмотрим пример использования Локальных переменных. Создадим структуру цикла For. Разместим внутри него генератор случайных чисел и функцию задержки. На передней панели создадим элемент управления Num Ctrl (для задания числа итераций цикла) и два элемента отображения Num Ind (рис. 20 а). Попробуем передавать данные, создаваемые внутри цикла (случайные числа и номер итерации), в элементы отображения которые не располагаются внутри цикла. Добавим в структуру цикла элемент Локальной переменной. Элемент находиться в подпункте Структуры (Structures) пункта Программирование основного меню в области программного кода. Первоначально Локальная переменная выглядит в виде квадрата со знаком вопроса внутри и имеет один входной терминал (рис. 20 а). Особенностью Локальной переменной является то, что она может быть привязана только к элементу ввода или отображения данных. Нажмем на элемент Локальной переменной левой кнопкой мыши, появиться меню с

названием имеющихся в программе элементов ввода и индикаторов[IV](рис. 20 а). Выберем пункт Numeric 2, что соответствует одному из индикаторов отображения. При этом Локальная переменная станет называться Numeric 2 и изменит свой цвет на желтый (рис. 20 б), что соответствует данным числового типа (в соответствии с типом данных индикатора Numeric 2).



а)

б)



в)

Рисунок 20. Локальные перемененные: а – добавление локальной переменной в структуру цикла, б – присвоение типа данных локальной переменной и привязка к элементу отображения, в – локальные переменные записи и чтения.

Соединим Локальную переменную с Numeric 2 генератором случайных чисел (рис. 20 б). Аналогично добавим в структуру цикла вторую Локальную переменную, свяжем ее с элементом отображения Numeric 3 и присоединим к

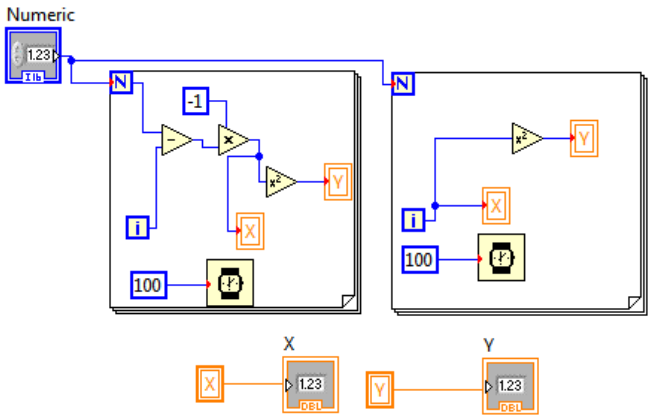
[IV] Название элементов ввода и индикаторов можно изменять, путем изменения расположенной над элементом подписи. Это возможно как на передней панели, так и в области программного кода.

счетчику итераций цикла (рис. 20 в). Теперь можно использовать созданы локальные переменные Numeric 2 и Numeric 3 в любом участке программного кода. Создадим еще две Локальные переменные и разместим их за пределами структуры цикла For. Аналогично свяжем их с индикаторами Numeric 2 и Numeric 3 однако сделаем эти переменные источниками данных. Для этого в меню, появляющемся по нажатию правой кнопки мыши на символ Локальной переменной, выберем пункт Change To Read (изменить на чтение). После этого Локальная переменная будет иметь выходной терминал, на котором будут появляться данные после вхождения их в Локальную переменную с таким же названием и находящуюся в режиме записи (в рассматриваемом примере Локальные переменные внутри структуры цикла). Соединим Локальные переменные вне цикла с индикаторами Numeric 2 и Numeric 3 (рис. 20 в). Теперь после запуска программы в индикаторах Numeric 2 и Numeric 3 будут отображаться текущие величины, генерируемые внутри цикла For.

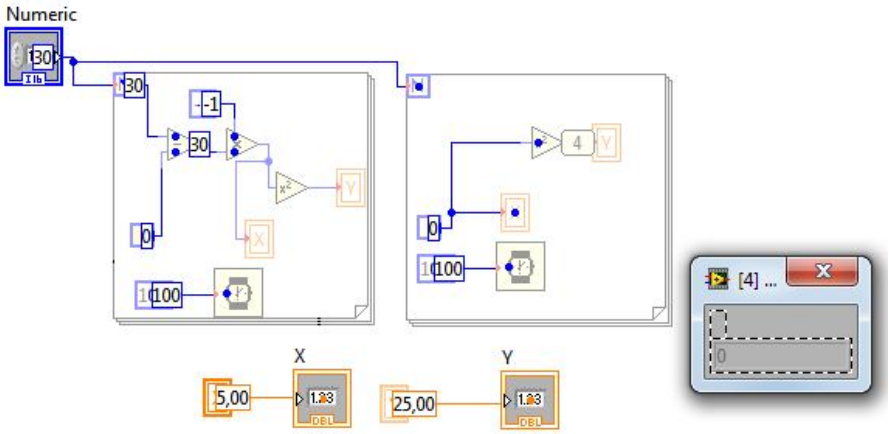
Рассмотрим пример использования Локальных переменных. Создадим программу, которая будет формировать данные для построения параболы. Будем задавать диапазон значений величины X, которая будет изменяться в диапазоне от –X до +X с шагом 1. Используем два цикла For. В первом цикле будем изменять X от заданной в элементе управления Numeric величины (со знаком -) до нуля. Во втором цикле величина X изменяется от нуля до заданной величины. В обоих циклах величина X возводиться в квадрат, что формирует данные для величины Y (рис. 21 а). В циклах использовались функции операций над числовыми данными, такие как вычитание, умножение и возведение в квадрат. Эти функции находятся в подпункте Numeric пункта Programming в области программного кода. В ходе работы циклов данные передаются в Локальные переменные записи X и Y. Также имеются соответствующие Локальные переменные чтения, которые соединены с элементами отображения X и Y (рис. 21 а). Запустим программу, например для 30 итераций цикла, задаваемых в элементе управления

Numeric. В индикаторах X и Y на передней панели будут отображаться соответствующие переменные, однако необходимой последовательности изменения аргумента и функции получено не будет. В частности аргумент X будет изменяться случайным образом, а не от -30 до 30 с шагом 1. Это связано с тем, что каждый элемент программного кода выполняется *сразу* после прихода в него необходимых данных по проводникам. Причем элементы программного кода выполняются независимо от других элементов. На это обращалось внимание ранее при рассмотрении структуры последовательности.

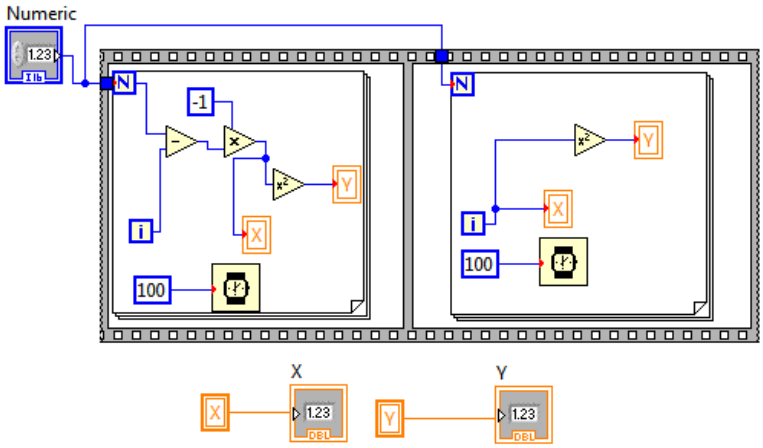
Движение потока данных в программе и последовательность исполнения ее элементов можно отследить в пошаговом режиме работы программы. Пошаговый режим полезен для отладки работы программы и ее оптимизации. Пошаговый режим включается только в области программного кода кнопкой с изображением лампы (Highlight Execution – подсветка выполнения), расположенной в ряду с кнопками запуска и принудительной остановки. В пошаговом режиме выполнение программы сильно замедляется. Не функционирующие в текущий момент элементы программы отображаются менее ярко по сравнению с обычным режимом (рис. 21 б). Задействованные элементы и проводники, по которым передаются данные, отображаются обычным образом. При этом отображаются значения переменных, подаваемых проводниками на терминалы элементов. Если нажать левой кнопкой мыши на проводник, то появиться отдельное окно (пробник) в котором будет отображаться значения переменных, передаваемых в проводнике (рис. 21 б).



а)



б)



в)

Рисунок 21. Пример возникновения неопределенности в порядке исполнения частей программного кода: а – отсутствует однозначное определение последовательности исполнения, б – включен режим отладки (пошаговое исполнение), в – последовательности исполнения определена однозанчно.

Посмотрим в этом режиме работу рассмотренной выше программы, состоящей из двух циклов For (рис. 21 а). Оказывается, хотя первый цикл For расположен левее второго, данные в Локальные переменные записи X и Y передаются сначала в правом цикле. Это связано с тем, что запуск циклов происходит практически одновременно (после поступления величины определяющей число итераций), однако во втором цикле For выполняется меньше операций над переменными, поэтому он исполняется быстрее. Для того, чтобы однозначно определить порядок исполнения элементов программного кода, нужно использовать структур последовательности, как было рассмотрено выше (рис. 16). Поместим первый цикл в первом кадре, а второй в следующем кадре (рис. 21 в). Теперь последовательность выполнения циклов однозначно определена, программа работает нужным образом - аргумент X изменяется от -30 до 30 с шагом 1 (что отображается в индикаторе X), а индикатор Y отображает квадрат величины X. Если будем отображать зависимость X от Y, то должна получаться парабола.

Попробуем визуализировать процесс получения двумерного массива данных, соответствующего различным точкам параболы, т.е. построим зависимость X(Y) в ходе выполнения циклов программы. Очевидно, что нужно использовать Сдвиговые регистры цикла, как было рассмотрено выше, и разместить виртуальный подприбор отображения XY Graph внутри цикла (рис. 19). Однако возникает вопрос, в каком цикле размещать подприбор отображения Build XY Graph с индикатором XY Graph? Подприбор Build XY Graph нужно будет разместить в обоих циклах, поскольку данные, необходимые для построения параболы, будут генерироваться в обоих циклах и необходимо формировать их них одномерные массивы для передачи во входные терминалы X/Y Input подприбора Build XY Graph (рис. 22 а). В то же время нельзя поместить индикатор XY Graph, который отображает поступающий из подприбора Build XY Graph двумерный массив данных, одновременно в обоих циклах, поскольку это означает создание двух отдельных индикаторов XY Graph на

передней панели программы. В этом случае нужно разместить один индикатор XY Graph в любом цикле, а в другом цикле создать Локальную переменную и присвоить ей значения индикатора XY Graph (рис. 22а). Особенностью Локальных переменных является то, что данные в них передаются не зависимо от того, в каком месте программного кода они располагаются и выполняется ли в настоящее время этот участок программного кода. Обратим внимание, что в рассматриваемом примере выходные терминалы Сдвиговых регистров первого цикла соединены со входными терминалам Сдвиговых регистров второго цикла. Это необходимо для того, чтобы массив данных, сформированный в первом цикле (половина зависимости X(Y)), была передана во второй цикл и использовалась для построения полной зависимости X(Y).

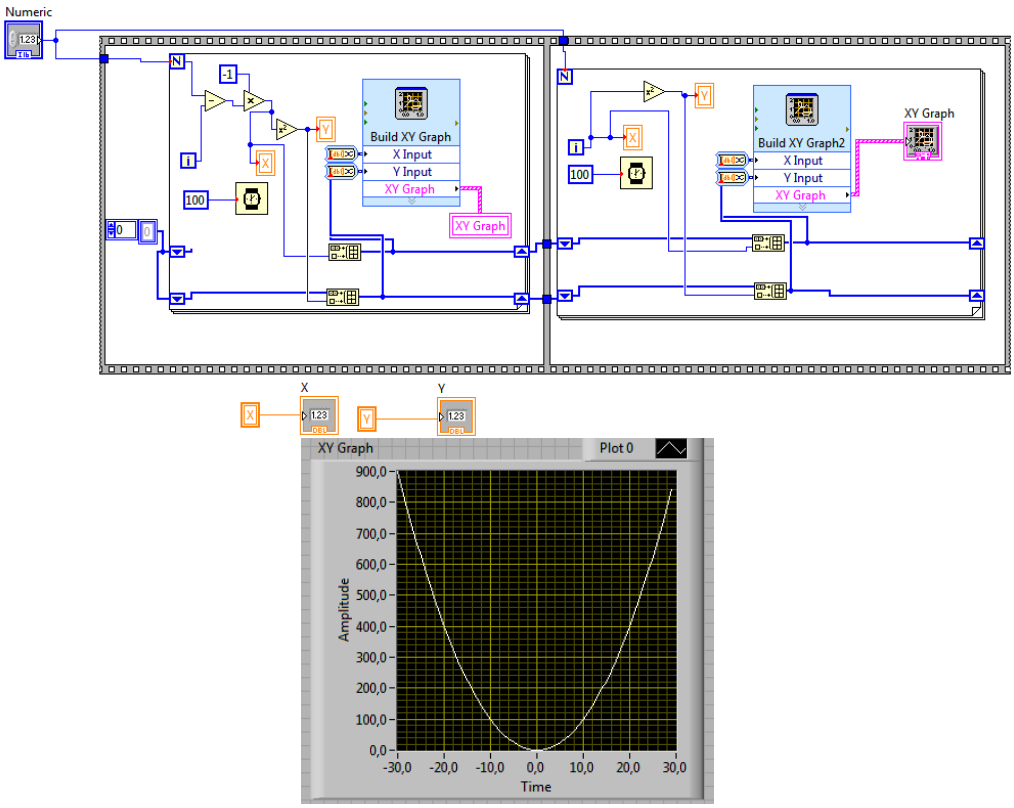
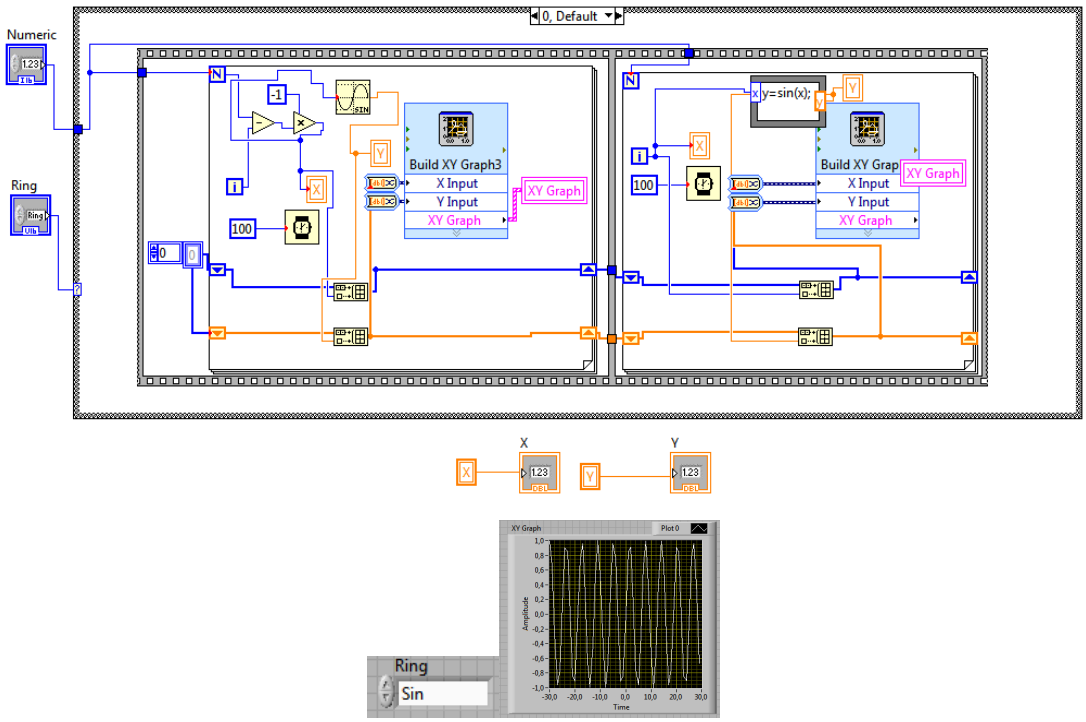
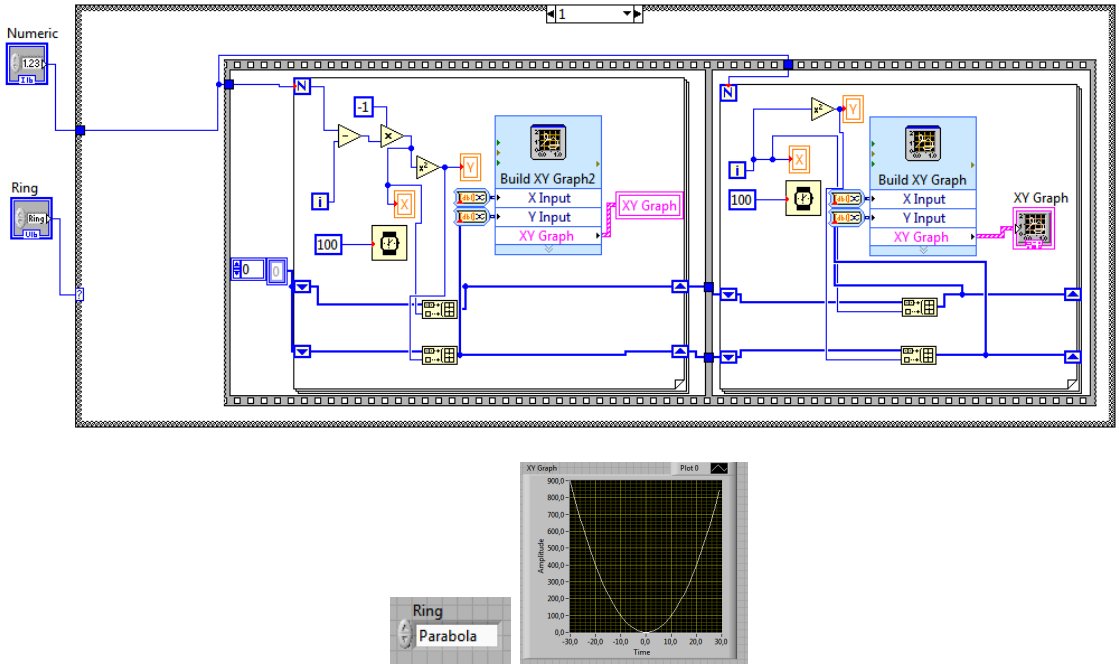


Рисунок 22. Использование локальной переменной для передачи данных двумерного массива.



а)



б)

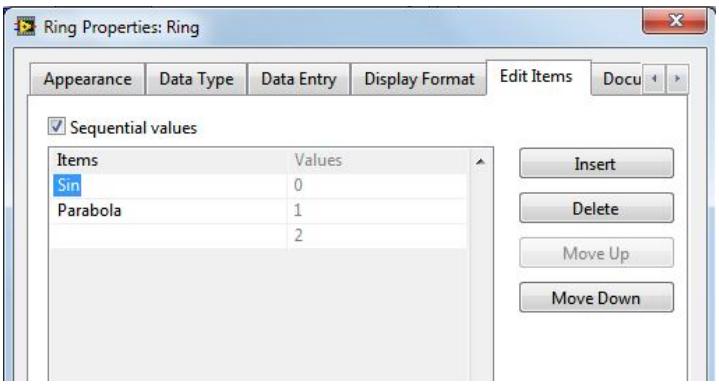
Рисунок 23. Использование Структуры выбора для организации альтернативных путей работы программы. а – Структура последовательности, расположенная в первом Случае Структуры выбора и часть передней панели, б – Структура последовательности, расположенная во втором Случае Структуры выбора и часть передней панели.

Рассмотренный пример может быть использован для автоматизации проведения физического эксперимента. Подобную структуру, например, можно использовать для управления установкой по получению зависимости сопротивления Холла от величины магнитного поля. В ходе работы цикла можно проводить задание напряжения на электромагните и практически одновременную регистрацию измеряемого сопротивления Холла. При этом будет проводиться отображение текущих значений измеряемых величин.

1.5. Структура выбора. Условный оператор.

Часто возникает необходимость в реализации различных алгоритмов работы программы в зависимости от получаемых результатов. Для этого можно использовать условный оператор If и Структуру выбора. Рассмотрим сначала работу со Структурой выбора. Создадим программу, которая будет строить зависимость $Y(X)=X^2$ (параболу, как в предыдущем примере), либо зависимость $Y(X)=Sin(X)$ в зависимости от того, что будет выбрано до запуска программы. За основу возьмем предыдущий пример. Создадим в программном коде Структуру выбора. Для этого в подпункте Structures пункта Programming выберем элемент Case Structure (Структура выбора). После этого очертим прямоугольную область вокруг Структуры последовательности. При этом будет создана Структура выбора, содержащая Структуру последовательности. По умолчанию Структура последовательности состоит из двух Случаев (Case) реализуемых при поступлении логической переменной на входной терминал структуры. В связи с этим Случаи называются True и False. Программный код для одного Случая может быть совершенно другим по сравнению со вторым Случаем. В Структуре выбора виден программный код, расположенный только в одном Случае. Для переключения между окнами Случаев служит панель вверху Структуры выбора (рис. 23).

Выбор Случая может осуществляться не только подачей логической переменной на входной терминал, но также подачей целочисленной константы, соответствующей номеру Случая, начиная с нулевого. Подобный выбор удобно делать с помощью элемента управления Ring. Элемент Ring расположен во вкладке Ring @ Enum в разделе Modern на передней панели программы. Элемент управления Ring позволяет выбрать событие, имеющее определенное название. Каждому событию будет соответствовать целочисленная константа, выходящая из терминала элемента Ring. Для редактирования названия события и присвоения ему константы нужно зайти в свойства данного элемента, путем выбора пункта Properties по нажатию правой кнопки мыши. События расположены в пункте Edit Items (рис. 24 а). В строчке Items можно ввести название события. В строчке Values видны соответствующие константы (рис. 24 а). Соединим элемент Ring с входным терминалом Структуры выбора. При этом название Случаи получат название 0 и 1 (рис. 23).



а)

б)

Рисунок 24. а – Окно редактирования событий элемента управления Ring, б – функцию Formula Node (первоначальная форма функции, добавление терминалов, редактирование алгоритма обработки данных).

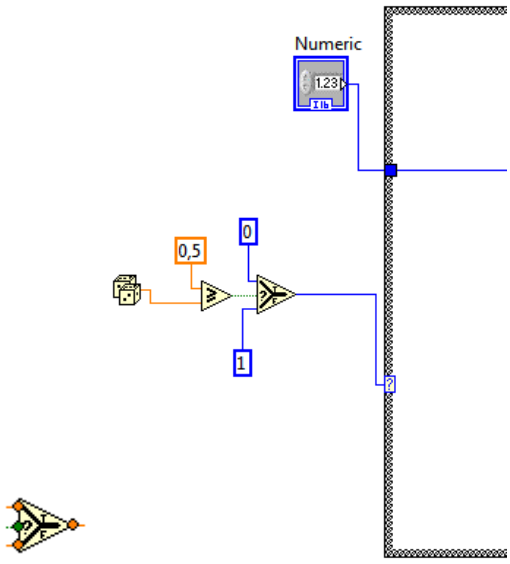
В одном из Случаев будет расположена Структура последовательности, которая была рассмотрена выше (рис. 22). Во втором Случае нужно создать аналогичную Структур последовательности. Обратим внимание на ряд особенностей. Элемент управления Numeric, задающий число итераций циклов должен быть один для циклов обоих Случаев. Поэтому его нужно вынести за пределы Структуры выбора и соединить проводником с соответствующими терминалами циклов в обоих Случаях (рис. 23). Индикатор XY Graph должен быть один. Если просто скопировать Структуру последовательности из одного Случая в другой, то на передней панели программы появиться второй индикатор XY Graph 2. Его нужно удалить. Также нужно проследить, чтобы Локальные переменные в обоих Случаях были одинаковыми в соответствующих местах. Т.е. не должны появиться Локальные переменные X2, Y2, XY Graph 2. Для Случая 0 программа должна строить зависимость Y(X)=Sin(X), где X номер итерации цикла. Для получения величины Sin(X) можно использовать встроенную функцию синуса. Функция расположена в подпункте Trigonometric Functions пункта Elementary вкладки Mathematics основного меню в области программного кода. Представляет собой квадратный элемент с изображением синуса, имеющий один входной и один выходной терминал (рис. 23 а, первый цикл).

В LabView возможно альтернативное задание функциональных зависимостей. Можно использовать функцию Formula Node (Узел формулы), расположенною в подпункте Structures пункта Programming в области программного кода. Первоначально Formula Node представляет собой прямоугольную область без терминалов (рис. 24 б). Для использования Formula Node нужно добавить входной терминал (терминалы) в который будут входить данные и выходной терминал (терминалы) из которого будут выходить данные, обработанные по указанному в Formula Node алгоритму. Для добавления терминалов нужно выбрать пункты Add Input и Add Output по нажатию правой кнопки мыши на границах Formula Node. Далее по

нажатию левой кнопки мыши на созданный терминал нужно в нем ввести название входящей (выходящей) переменной (рис. 24 б). Назовем входную перемену x, а выходную y. Теперь внутри Узла напишем алгоритм обработки входной переменной. В нашем случае это будет y=sin(x) (рис. 24 б). Функция Formula Node нужна для возможности компактной записи сложных математических формул. Также внутри Formula Node можно размещать даже программный код, написанный в стиле языка программирования С.

Теперь в рассматриваемой программе можем выбрать, какое Событие программного кода будет реализовано – построение параболы или построения синуса. После запуска программы будет выполнено соответствующие Событие (рис 23 а и б).

Теперь используем для определения реализации События условный оператор If. Условный оператор If является одной их основных функция любого языка программирования.



а) б)

Рисунок 26. а – Графический элемент условного оператора If, б – Использование условного оператора для определения реализации События Структуры выбора.

Графический элемент, соответствующий условному оператору в языке программирования G, находиться в области программного кода в подпункте операций сравнения (Comparison) пункта Программирование (Programming). Оператор отображается в виде треугольника имеющего три входных терминала и один выходной (рис. 26 а). Центральный терминал (s) соответствует данным логического типа. При поступлении на центральный терминал логической константы True на выходной терминал оператора будет передано значение, поступившее на верхний входной терминал (t). Соответственно при поступлении константы False на выходной терминал передается значение, поступившее на нижний входной терминал (f). Добавим в ранее рассмотренную программу часть, содержащую условный оператор и оператор сравнения ≥ (также расположен в Comparison), генератор случайных чисел и числовые константы (0.5, 0 и 1). Соединим добавленные элементы таким образом, чтобы после генерации случайного числа проводилась его сравнение с константой 0.5 и в случае если число ≥ 0.5, условный оператор должен выдавать константу 0 (рис. 26 б). Теперь после запуска программы будет отображаться зависимость Y(X)=Sin(X) или $Y(X)=X^2$ в зависимости от случайной величины.

2. Удаленное управление приборами с использованием программной среды LabView.

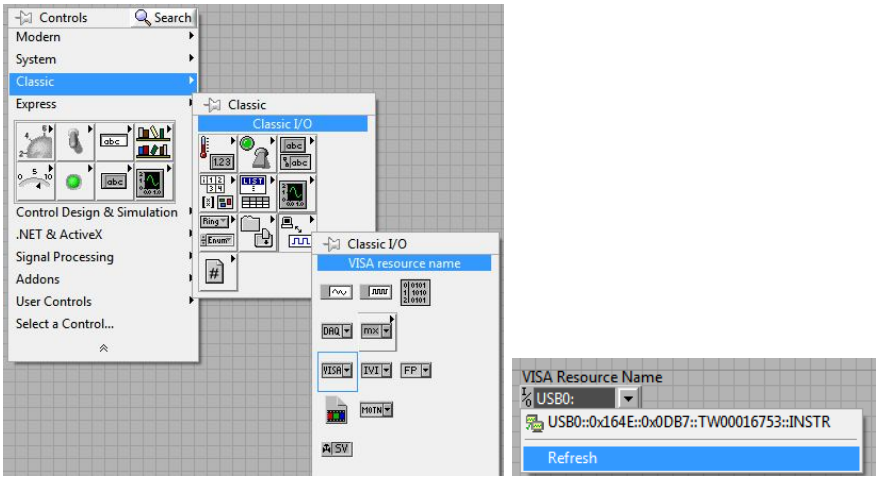
2.1. Программный интерфейс VISA. Функции обмена данными с прибором.

Используя рассмотренные основы программирования в LabView, организуем получение и сохранение данных с использованием мультимета АКИП В7-78. Для взаимодействия программной среды LabView с приборами необходимо на управляющий компьютер установить пакет программных драйверов интерфейса VISA (Virtual Instrument Software Architecture – архитектура программных виртуальных инструментов). Набор программных инструментов, содержащихся в пакете драйверов VISA, позволяет организовывать с помощью среды LabView управление приборами различных производителей и имеющие различные коммуникационные интерфейсы. Полностью совместимые со средой LabView пакеты драйверов VISA можно бесплатно загрузить с сайтов производителей оборудования и программного обеспечения National Instruments (NI-VISA[1]), Agilent Technologies (Agilent IO Libraries Suite[2]), Keithley Instruments (Keithley I/O Layer[3]).

Рассматриваемый мультиметр АКИП В7-78 имеет интерфейс USB и является типичным представителем приборов такого класса. Приборы с интерфейсом USB требуют установки дополнительного пакета драйверов поставляемого производителем прибора. Такой пакет драйверов является самостоятельным программным обеспечением не связанным с LabView и пакетом драйверов VISA. После установки драйверов интерфейса USB, прибор отображается в виде Диспетчере устройств Windows в виде устройства USB с соответствующим названием. Теперь прибор готов для управления с помощью среды LabView.

Разместим на передней панели элемент управления, определяющий (выбирающий) аппаратный интерфейс к которым будет работать программа. Нажатием правой кнопки мыши вызовем основное меню объектов (Controls) и далее во вкладке Classic I/O пункта Classic выберем элемент VISA resource

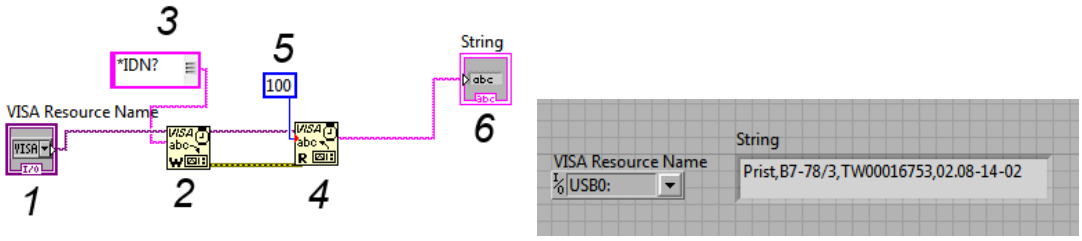
name (имя ресурса VISA) (рис. 27 а). Этот элемент отображается на передней панели в виде выпадающего меню, в котором отображены номера коммуникационных интерфейсов, с которыми может быть установлена связь посредством драйверов VISA.



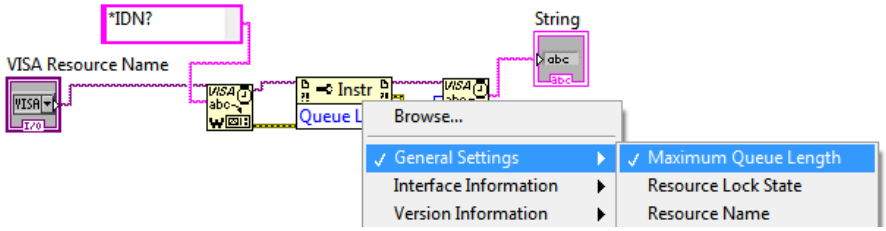
а) б)

Рисунок 27. а – меню элементов ввода-вывода, б – элемент выбора коммуникационного интерфейса.

В рассматриваемом примере появиться номер и адрес интерфейса USB к которому подключен мультиметр (рис. 27 б). В окне программного года этому элементу управления соответствует выходной терминал с названием VISA Resource Name (рис. 28 а, элемент 1). Из этого выходного терминала выходит массив данных, содержащих описание характеристик используемого интерфейса. Программисту обычно нет необходимости знать содержимое этого массива. Такой массив передается от одного элемента участка программного кода, отвечающего за взаимодействие с прибором, к другому элементу до завершения программы взаимодействия с прибором. Массив данных от элемента VISA Resource Name должен проходить через все участки программного кода в которых происходит обращение к прибору.



а)



б)

Рисунок 28. а – Программа отправки запроса и получения ответа и часть передней панели, б – передняя панель программы

Теперь прибору можно передавать команды и считывать получаемый ответ. Для передачи команды прибору используется функция VISA Write (запись в VISA). Функция располагается в меню основных функций области программного кода по следующему адресу Instrument I/O (инструмент ввода/вывода) –> VISA –> VISA Write. Данная функция в окне программного кода отображается в виде белого квадрата с входными и выходными терминалами (рис. 28 а, элемент 2). К верхнему входному (слева) терминалу элемента VISA Write необходимо подвести поток данных их элемента VISA Resource Name, определяющих адрес физического интерфейса прибора. Из верхнего выходного (справа) терминала будет выходить такой же по типу поток данных для следующего элемента программы, отвечающего за взаимодействие с прибором.

По стандарту интерфейса VISA передача команд в прибор осуществляется строковым типом данных в кодировке ASCII, прибор также передает ответ кодировке ASCII. Рассмотрим передачу в мультиметр команду, являющуюся запросом об идентификационных данных. Команда *IDN? является стандартной для приборов, поддерживающих набор команд

SCPI (Standard Commands for Programmable Instruments -стандартные команды для программируемых инструментов). Полный набор SCPI команд, поддерживаемый конкретным прибором указывается в инструкции по эксплуатации прибора или в инструкции по его программированию. Команда *IDN? оканчивается знаком вопроса, согласно стандарту SCPI прибор должен выдать ответ. Ответом на данную команду является массив строковых данных, содержащих информацию о названии прибора, его номере и др. Простейшим способов передачи команды в прибор является использование строковой константы. Для этого в области программного кода в меню основных функций по адресу Programming –> String (меню функций и команд работы со строковым типом данных) –> String Constant (Строковая константа) нужно выбрать символ строковой константы представляющей собой прямоугольник розового цвета. В область строковой константы может быть вписана команда, передаваемая прибору (рис. 28 а, элемент 3). Строковая константа имеет выходной терминал, который должен быть соединен с входным терминалом розового цвета (write buffer) функции VISA Write (рис. 28 а). После запуска программы данная команда будет передана в прибор, и он будет готов для передачи ответа.

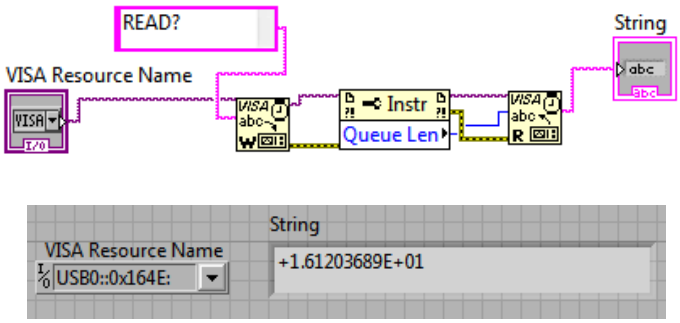
Следует отметить, что Строковая константа имеет несколько режимов ввода текста. Наиболее часто используемым режимом является режим Normal Display. Режим можно выбрать в меню, появляющимся после нажатия провой кнопки мыши на Строковую константу. В этом виде не отображаются специальные командные символы, соответсвующие, например, пробелу или команде перехода на новую сроку. В большинстве случаев команды SCPI можно вводить в режиме Normal Display. Однако для некоторых приборов и команд нужно внимательно следить за структурой, отправляемой в прибор строки, особенно если она состоит из нескольких команд. Обычно отправляемые команды должны оканчиваться специальными символами \n (переход на новую сроку) и \r (переход к началу строки). Эти символы в режиме Normal Display, но становиться видимыми в

режиме `\` Code Display. Если возникают проблема с принятием прибором команды (выражается обычно в появлении надписи Err на дисплее прибора), то нужно перейти в режим `\` Code Display для Строковой константы и проверить структуру команды.

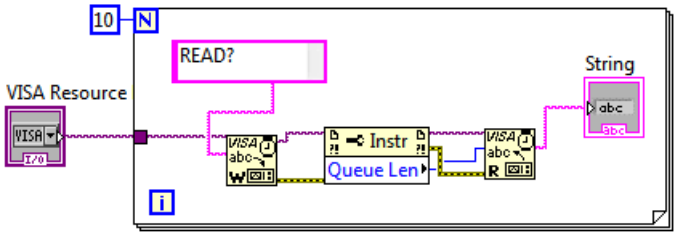
Для получения ответа, отправляемого прибором после обработки поступившей команды, используется функция VISA Read (рис. 28 а, элемент 4), предназначенная для считывания определенного объема данных из буфера обмена прибора. Входной терминал функции VISA Read, соответствующий потоку данных из элемента VISA Resource Name, должен быть соединен с соответствующим элементом функции VISA Write (рис. 28 а). В этом случае функция VISA Read (как и Write) направлена на взаимодействие именно с конкретным прибором, поскольку одновременно программа может работать с несколькими приборами, в том числе имеющими один тип коммуникационного интерфейса (но с разными адресами). Функции работы с VISA имеют также канал передачи информации об ошибки, возникающей при обращении к прибору (например, ошибке, связанной с невозможность обращения к буферу прибора). Терминалы этого канала расположены внизу функций VISA Write, Read и других, работающих с линией VISA (рис. 28 а). Этот канал данных не является обязательным. Считывание данных из буфера прибора происходит в тот момент, когда поток данных линии, связанной с элементом VISA Resource Name дойдет из функции VISA Write в функцию VISA Read (рис. 28 а). Для считывания данных нужно указать объем в байтах, какой нужно получить из буфера обмена. Для этого к входному терминалу byte count функции VISA Read нужно подсоединить целочисленную константу, определяющий размер считываемых данных. Для рассматриваемого примера этот размер взят с запасом (больше нужного) (рис. 28 а, элемент 5). Если запрашиваемый объем данных меньше занимаемого в буфере обмена, то будет считана только часть ответа прибора. В большинстве случаев объем данных в буфере обмена может быть определен автоматически. Для этого

используется функция VISA Property Node (Узел свойства VISA). Функция расположена в области программного кода по следующему адресу Instrument I/O –> VISA –> VISA Advanced –> VISA Property Node. Первоначально функция появляется в виде шаблона имеющего входные и выходные терминалы линии VISA и линии ошибок, а также имеется выходной терминал для выхода данных различного типа. Общее функциональное назначение VISA Property Node является сообщение различной информации о используемой линии VISA и используемом интерфейсе. Конкретное назначение функции задается путем выбора нужного пункта в меню, появляющемся при нажатии левой кнопки мыши на графический символ функции. Для нашей задачи необходимо выбрать подпункт Maximum Queue Length (максимальная длина очереди) пункта General Settings (основные установки) (рис. 28 б). Теперь функция Maximum Queue Length будет выдавать объем данных доступных в буфере обмена, что может использоваться для функции VISA Read (рис. 28 б).

Поскольку считанная информация имеет строковый тип данных, то для ее вывода на переднюю панель нужно использовать индикатор строковых данных. В рассмотренном в пункте 1.1. примере генерации и вывода данных использовался размещенный на передней панели индикатор числовых типов данных Num Ind. В рассматриваемом примере необходимо разместить на передней панели индикатор данных строкового типа String Ind, который располагается в пункте Express меню Controls на передней панели. После размещения индикатора на передней панели, в области программного кода возникнет элемент String соответствующий индикатору (рис. 28 а, элемент 6). Элемент String имеет входной терминал, который нужно соединить с выходным терминалом read buffer функции VISA Read. В этот входной терминал индикатора поступит массив данных строкового типа, считанный из буфера обмена прибора. После запуска программы в индикаторе на передней панели появиться информация о приборе (рис. 28 б).



а)



б)

Рисунок 29. а – Программа отправки запроса в прибор и получения ответа и часть передней панели, в – дистанционное измерение 10 значений сопротивления.

Запросим у прибора текущее значение измеряемой величины. Для этого с передней панели прибора переведем мультиметр в режим измерения сопротивления и подключим к измерительным терминалам прибора резистор. Протокол SCPI предусматривает несколько вариантов запроса измеряемой величины, различающиеся процессом проведения измерения и сохранением результата в буфер обмена прибора. Наиболее простым является отправка команды READ?, которая запрашивает текущее значение измеряемой величины. После получения команды прибор помещает текущее значение в буфер, откуда его можно считать с помощью функции функцию VISA Read. После очередного получения команды READ? значение измеренной величины в буфере обмена заменяется новым. На рис. 29 а представлена передняя панель программы с отображением в индикаторе String Ind величины сопротивления, измеренного мультметром.

Очевидно, что для автоматического считывания множества измеряемых прибором значений нужно использовать цикл. Используем цикл с фиксированным числом итераций For, как рассматривалось выше. В результате получим программу для автоматического измерения заданного числа значений сопротивления (рис. 29 б). Программа аналогична рассмотренной выше программе для генерации заданного числа случайных чисел (рис. 3), но теперь происходит автоматическое измерение реальной физической величины (сопротивления). Аналогично можно использовать цикл по условию While.

2.2. Преобразование типа данных.

В рассмотренном примере в индикатор String Ind на передней панели программы выводиться полученные с прибора значение в виде переменной строкового типа. Для операций со значениями измеренной величины необходим перевод данных из строкового типа данных в числовой. Для этого в LabView имеется ряд функции, расположенных в области программного кода в подпункте String пункта Programming. В рассматриваемом случае прибор выдает данные в простой форме, в виде экспоненциальной записи числа. Например, измеренное сопротивление в 16.12 Ом выводиться в виде +1.612E+01 (рис. 29 а). В LabView имеется специальная функция Fract/Exp String To Number Function для перевода строковой переменной такого типа в численную переменную. Функция находится по адресу Programming –> String –> String/Number Conversion –> Fract/Exp String To Number Function.

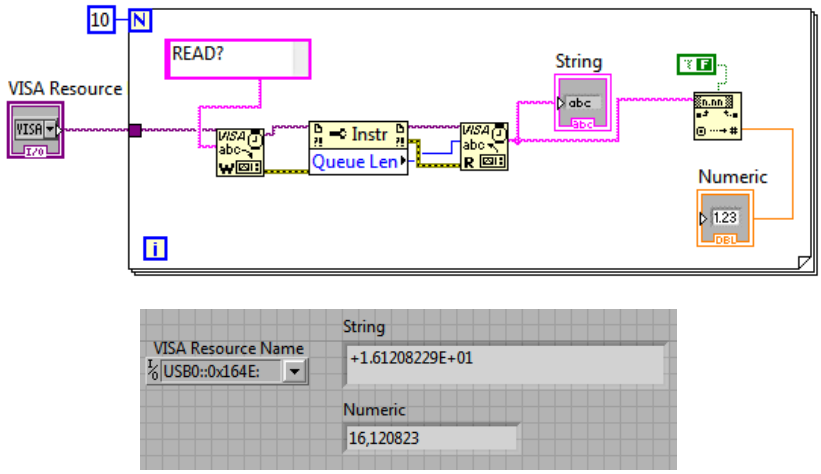
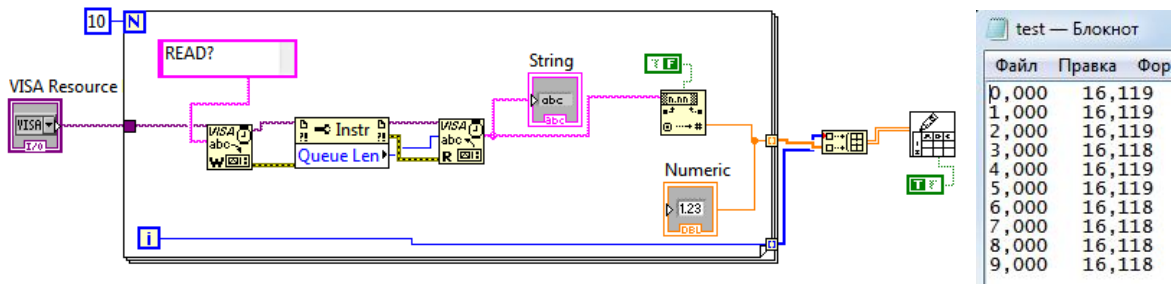


Рисунок 30. Программа отправки запроса в прибор и получения ответа с конвертацией типа данных, а также часть передней панели.

Функция Fract/Exp String To Number Function имеет входной терминал string для переменной строкового типа и выходной терминал number для переменной числового типа. Подсоединим выходной терминал функции VISA Read с данными, полученными от прибора, к терминалу string функции Fract/Exp String To Number Function, а для отображения получаемого числа используем индикатор Num Ind на передней панели (рис. 30). Функция Fract/Exp String To Number Function имеет входной терминал логического типа use system decimal point (использовать системный десятичный разделитель). По умолчанию принимается состояние на входе этого терминала как True, и функция преобразования типа данных использует в качестве десятичного разделителя тот, что установлен в настройках операционной системы. Однако, как правило, в русифицированной ОС Windows десятичным разделителем является запятая. В рассматриваемом примере прибор выдает данные с точкой в качестве десятичного разделителя (рис. 30). Поэтому на вход терминала use system decimal point функции преобразования типа данных нужно подать логическую константу False. В этом случае преобразование типа данных проводится правильно и на выходе функции появляется переменная числового типа данных (рис. 30).



а) б)

Рисунок 31. а – код программы измерения 10 значений сопротивления, б – сохраненный текстовый файл с результатами измерений.

При использовании цикла, после преобразования типа данных[V] возможно сохранение полученных данных в массив цикла (индексация) с последующим сохранением в файл. На рис. 31 представлена программа для измерения 10 значений сопротивления и текстовый файл содержащий результаты измерения. Также возможно создание сдвигового регистра и накопление данных внутри цикла с последующей передачей в Локальные переменные или в какой-либо индикатор, как было рассмотрено в главе 1.

[V] Накопление результатов и их сохранение возможно также в строковом типе данных.

3. Пример автоматизации экспериментальной установи с использованием среды LabView.

Рассмотрим структуру программы управления установкой для исследования намагниченности методом переменного градиента силы. В установке используется источник-измеритель Keithley 2400, фазочувствительный вольтметр Stanford SR 810, мультметр АКИП В7-78.

Схематическое изображение установки представлено

а рис. 33. Установка состоит из измерительной головки с пьезоэлектрическим элементом (1), катушек градиентного переменного магнитного поля (2), электромагнита (3), датчика Холла (4), усилителя низкой частоты (5), фазочувствительного вольтметра, мультиметра для измерения ЭДС Холла (7), источника питания электромагнита (8), управляющего компьютера (9).

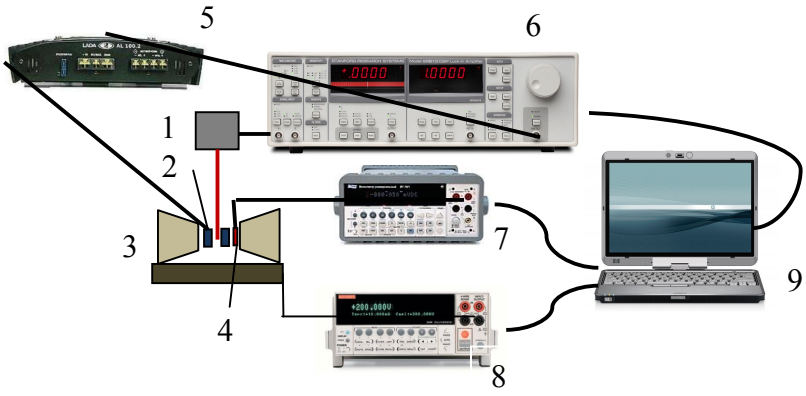


Рисунок 33. Структурная схема установки для измерения намагниченности.

С генератора низкой частоты фазочувствительного вольтметра поступает синусоидальный сигнал на усилитель низкой частоты. С усилителя низкой частоты синусоидальное напряжение подается на модулирующие катушки, создающие переменный градиент магнитно поля. Под действием переменного градиента магнитно поля магнитный образец на держателе начинает колебаться, что вызывает появление переменного напряжения на

выходе измерительной головки. Это напряжение регистрируется фазочувствительным вольтметром. Величина напряжения, создаваемого пьезоэлектрическим элементом, пропорциональна намагниченности исследуемого образца. Намагничивание образца происходит электромагнитом, который питается от программируемого источника питания. Величина намагничивающего магнитного поля (определяющая ЭДС Холла) регистрируется мультиметром. В процессе работы установки проводиться развертка величины магнитного поля от нуля до максимального значения для двух знаков полярности. При этом происходит регистрация напряжения, возникающего на пьезоэлементе. Следовательно, регистрируется зависимость, пропорциональная намагниченности исследуемого образца, от величины магнитного поля.

В целом процесс получения магнитополевой зависимости измеряемой величины выглядит следующим образом:

1. Задается максимальное значение магнитного поля, определяемого максимальной величиной тока, пропускаемого через электромагнит. Задается величина шага изменения пропускаемого тока, следовательно, задается шаг изменения магнитного поля;
2. Измеряется текущее значение магнитного поля путем регистрации величины ЭДС Холла и пересчета (с учетом калибровочной зависимости) в единицы измерения магнитного поля.
3. Измеряется величина напряжения на пьезоэлементе;
4. Ток, пропускаемый через электромагнит, изменяется на величину заданного шага по току. Повторяются пункты 2 и 3.
5. При достижении максимальной заданной величины магнитного поля (пропускаемого через электромагнит тока), повторяются пункты 2, 3, 4, но с уменьшением величины магнитного поля от максимального значения до нуля;
6. Повторение пунктов 2 – 5, но с противоположным знаком тока, пропускаемого через электромагнит;

7. Сохранение полученного массива данных, содержащего значения величины магнитного поля и измеряемой величины.

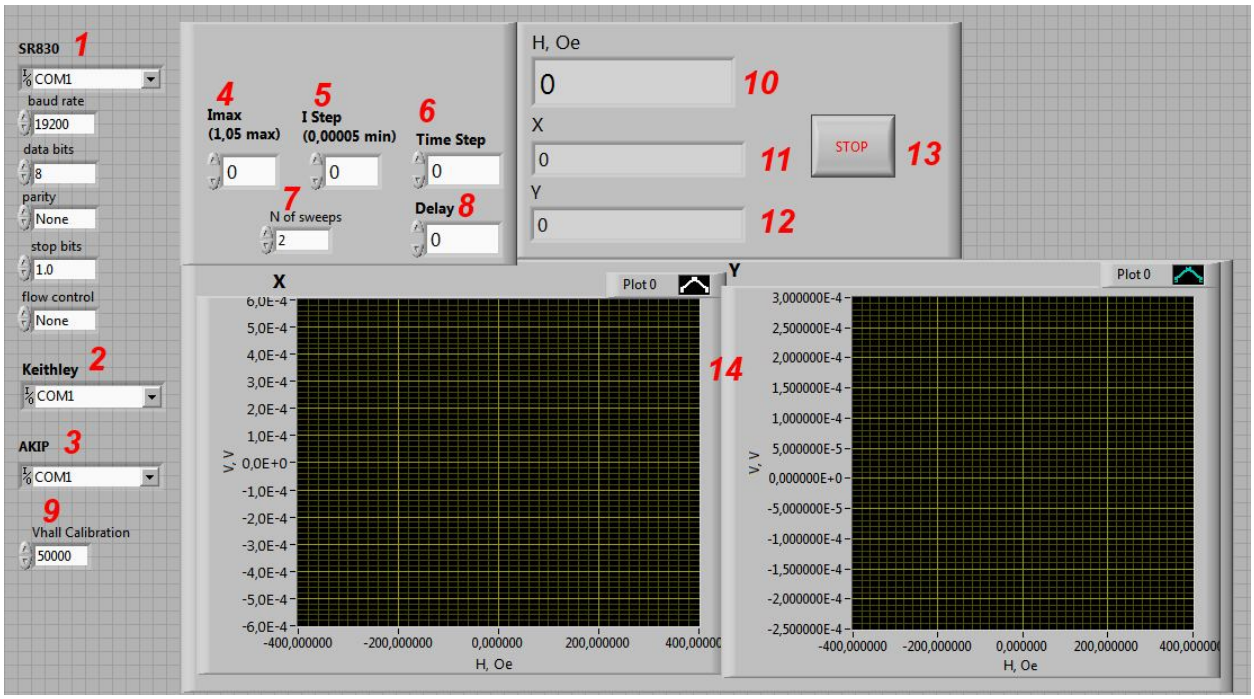


Рисунок 34. Передняя панель программы управления установкой исследования намагниченности методом переменного градиента силы.

На рисунке 34 представлена передняя панель программы управления установкой. На передней панели расположены:

1 – Область настройки параметров управления фазочувствительным вольтметром Stanford SR 810. Область состоит из элемента выбора аппаратного интерфейса VISA resource name, элементов ввода численных данных Num Ctrl и элементов Ring. Элементы Num Ctrl определяют параметры COM пора baud rate и data bits. Эти величины в явном виде передаются в функцию (виртуальный подприбор) VISA Configure Serial Port VI, определяющий взаимодействие с COM портом. Элементы Ring передают в элемент VISA Configure Serial Port числовые константы, соответствующие другим настройкам COM порт;

2 – Область настройки параметров управления источником питания электромагнита Keithley 2400. На передней панели присутствует только элемент VISA resource name. Параметры COM порта расположены в области программного кода в виде числовых констант и на переднюю панель не выведены.

3 – Область настройки параметров управления мультиметром АКИП В7-78. Состоит только из элемента VISA resource name;

4 – Элемент ввода численных данных Num Ctrl для задания величины максимального тока, пропускаемого через электромагнит;

5 – Элемент ввода численных данных Num Ctrl для задания шага по току, пропускаемому через электромагнит;

6 – Элемент ввода численных данных Num Ctrl для задания временной задержки между шагами по току (задержка необходима ввиду того, что электромагнит обладает значительной индуктивностью, и время установления создаваемого магнитного поля после изменения величины тока через магнит может составлять несколько сотен миллисекунд);

7– Элемент ввода численных данных Num Ctrl для задания числа разверток магнитного поля;

8 – Элемент ввода численных данных Num Ctrl для задания временной задержки между командой считывания значений с фазочувствительного вольтметра и началом считывания значений. Задержка необходима ввиду относительно малой скорости передачи данных через COM порт;

9 – Элемент ввода численных данных Num Ctrl, определяющий параметры калибровочной зависимости по пересчету ЭДС Холла с датчика Холла в эрстеды;

10 – Элемент отображения численных данных Num Ind, отображающий текущее значение магнитного поля;

11 – Элемент отображения численных данных Num Ind, отображающий текущее значение амплитуды напряжения на пьезоэлементе, синфазного с переменным напряжением на модулирующих катушках;

12 – Элемент отображения численных данных Num Ind, отображающий текущее значение амплитуды напряжения на пьезоэлементе, сдвинутого на 90 градусов по фазе относительно переменного напряжения на модулирующих катушках;

13 – Кнопка остановки stop button для остановки цикла While, отвечающего за накопления массива регистрируемых величин и их отображение в индикаторах XY Graph.

14 – Индикаторы XY Graph.

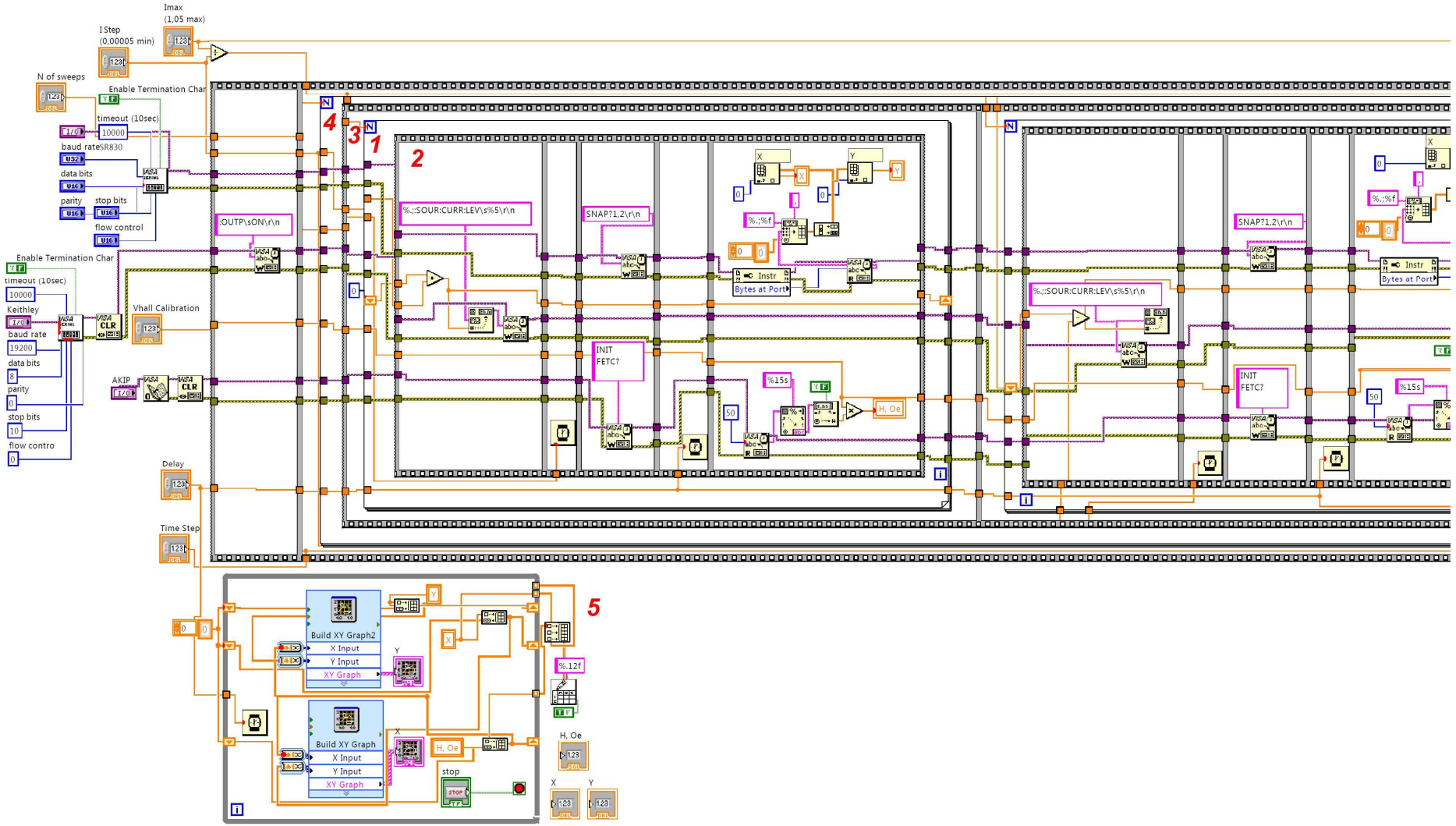
Рассмотрим основные элементы программы. Программный код представлен на рис. 35. Отметим, что программа по своей сути является развитием программы, рассмотренной в пункте 1.4 первой главы (рис. 22). Основным элементом программы является цикл For (1 на рис. 35). Число итераций цикла равно числу шагов по магнитному полю, следовательно, по току через магнит. Число шагов определяется как максимальная величина тока (а амперах), деленная на величину шаг по току (в амперах). Внутри цикла For расположена Структура последовательности (2 на рис. 35). В первом кадре последовательности с помощью функции VISA Write на источник питания магнита подается команда генерации тока. Величина тока определяется введенным значением шага по току и номером шага. В процессе работы цикла происходит суммирование значений шагов по току от итерации к итерации, от нулевого значения до максимального введенного значения тока. Для формирования команды в прибор используется функция Format Value Function, которая к шаблону команды в виде строковой переменной добавляет числовую переменную шага по току и конечную команду (в виде строковой константы) передает в функцию VISA Write. Во втором кадре последовательности находится функция задержки Wait для установки величины магнитного поля. В третьем кадре в линии VISA вольтметра и мультиметра посредством VISA Write передаются команды на передачу измеренных величин. В четвертом кадре находится функция задержки Wait для ожидания передачи данных по линии COM порта из

вольтметра. В пятом кадре происходит получение данных из вольтметра и мультметра. Данные из вольтмера содержат величину синфазного и квадратурного сигнала, поэтому передаются в виде одномерного массива данных строкового типа. Для преобразования данных в числовой массив используется функция Spreadsheet String To Array Function. Полученные данные передаются в локальные переменные считывания с названием X, Y, H,

Программа состоит из четырех рассмотренных циклов For. Первый цикл увеличивает ток через магнит от нуля до максимального значения. Второй цикл уменьшает ток от максимального значения до нуля. Второй и третий циклы повторяют первый и второй, но для противоположного знака величины тока через магнит (используется биполярный источник тока). Эти четыре цикла позволяют получить полную развертку по магнитному полю и при этом регистрировать измеряемые величины. Циклы находятся в Структуре последовательности (3 на рис. 35), определяющей последовательность выполнения циклов. Эта структура находится в цикле For (4 на рис. 35) который определяет число разверток по магнитному полю.

Регистрируемые данные через локальные переменные передаются в отдельно расположенный цикл While. В трех Сдвиговых регистрах этого происходит накопление данных и их отображение с использованием двух XY Graph. Сохранение данных производиться после остановки цикла While.

Программа работает с тремя отдельными приборами. Два имеют интерфейс COM, один интерфейс USB. Соответственно программа имеет три линии программного интерфейса VISA. Для корректной работы приборов на линии COM может требоваться функция VISA Clear Function, очищающая буфер обмена прибора. Функцию желательно использовать в начале и в конце программы. Для работы с приборами также желательно использовать функцию открытия сессии VISA (VISA Open Function) и функцию закрытия сессии (VISA Close Function).


Imax
(1,05 max)
I Step
(0,00005 min)
N of sweeps
Enable Termination Char
timeout (10sec)
10000
baud rate
SR830
data bits
parity
stop bits
flow control
:OUTP\sON\r\n
%.;:SOUR:CURR:LEV\s%5\r\n
SNAP?1,2\r\n
%.;%f
Instr
Bytes at Port
INIT
FETC?
%15s
H, Oe
50
Enable Termination Char
timeout (10sec)
10000
Keithley
baud rate
19200
data bits
8
parity
0
stop bits
10
flow contro
0
Vhall Calibration
AKIP
Delay
Time Step
1
2
3
4
5
Build XY Graph2
X Input
Y Input
XY Graph
Build XY Graph
%.12f
stop

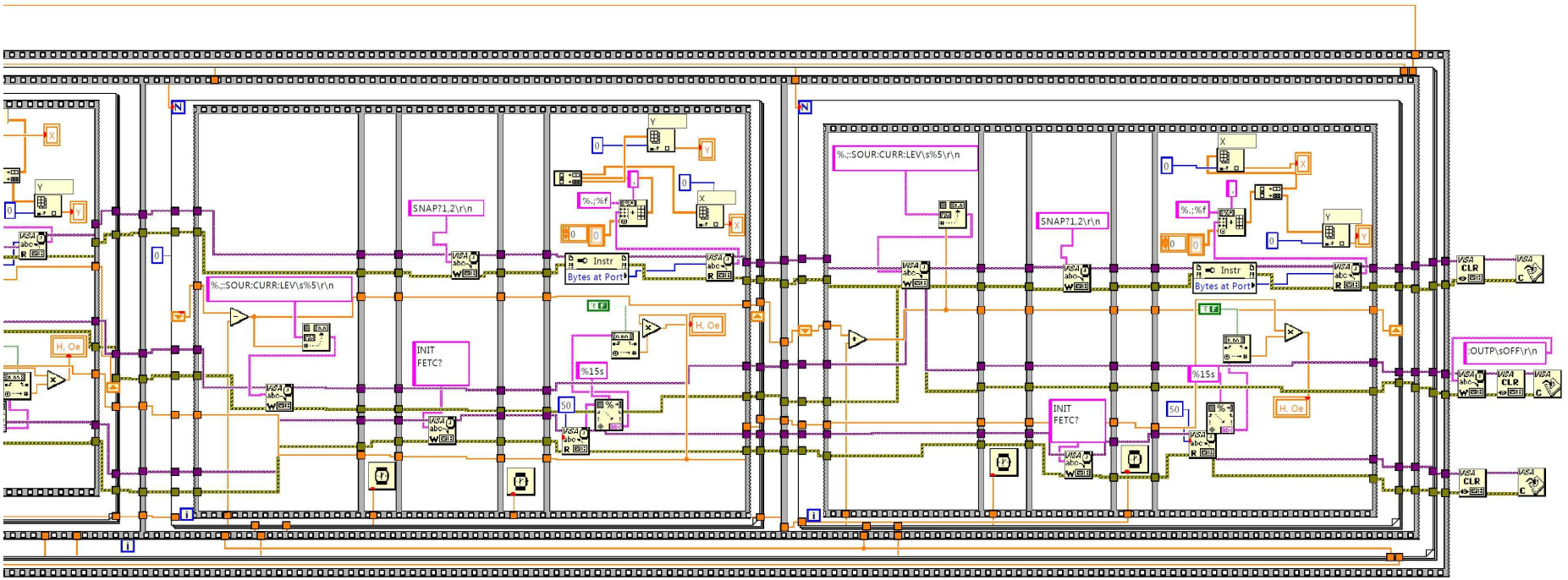


Рисунок 35. Код программы управления установкой исследования намагниченности методом переменного градиента силы

Заключение

Таким образом, в методическом пособии в доступной форме представлена информация, позволяющая легко организовать дистанционное управление измерительными приборами и сохранение измеренных величин. Обсуждение особенностей программ для управления действующими экспериментальными установками облегчает создание собственной программы по автоматизации процесса эксперимента. Для освоения представленного материала достаточно основных знаний по программированию.

Методическое пособие может быть использовано студентами старших курсов, магистрантов при выполнении научной части курсовой и дипломной работы. В частности студентам физического факультета студентов ННГУ, обучающимся по направлениям подготовки 210100 «Электроника и наноэлектроника», 222900 «Нанотехнологии и микросистемная техника». Методическое пособие будет полезно аспирантам, преподавателем и научным сотрудникам, занимающимся экспериментальной физикой. Также может быть использовано преподавателями для подготовки обновленных лабораторных работ по курсам «Физика полупроводников», «Основы радиоэлектроники», «Мехатроника и микроэлектромеханика», «Методы анализа и контроля наноструктурированных материалов и систем», «Физика твердого тела» с использованием современного оборудования закупленного по программе НИУ. Пособие будет полезно сотрудниками других факультетов работающих с экспериментальным оборудованием.

---

1 http://joule.ni.com/nidu/cds/view/p/id/3823/lang/en

2http://www.home.agilent.com/upload/cmc_upload/All/Agilent_IO_Libraries_Suite_16_2.htm?&cc=RU&lc=rus

3 http://www.keithley.com/support/data?asset=52766